EAI/Springer Innovations in Communication and Computing

Rashid Mehmood
Simon See · Iyad Katib
Imrich Chlamtac *Editors*

Smart Infrastructure and Applications

Foundations for Smarter Cities and Societies

EAI
RESEARCH MEETS INNOVATION

Springer

EAI/Springer Innovations in Communication and Computing

Series editor
Imrich Chlamtac, European Alliance for Innovation, Gent, Belgium

Editor's Note

The impact of information technologies is creating a new world yet not fully understood. The extent and speed of economic, life style and social changes already perceived in everyday life is hard to estimate without understanding the technological driving forces behind it. This series presents contributed volumes featuring the latest research and development in the various information engineering technologies that play a key role in this process.

The range of topics, focusing primarily on communications and computing engineering include, but are not limited to, wireless networks; mobile communication; design and learning; gaming; interaction; e-health and pervasive healthcare; energy management; smart grids; internet of things; cognitive radio networks; computation; cloud computing; ubiquitous connectivity, and in mode general smart living, smart cities, Internet of Things and more. The series publishes a combination of expanded papers selected from hosted and sponsored European Alliance for Innovation (EAI) conferences that present cutting edge, global research as well as provide new perspectives on traditional related engineering fields. This content, complemented with open calls for contribution of book titles and individual chapters, together maintain Springer's and EAI's high standards of academic excellence. The audience for the books consists of researchers, industry professionals, advanced level students as well as practitioners in related fields of activity include information and communication specialists, security experts, economists, urban planners, doctors, and in general representatives in all those walks of life affected ad contributing to the information revolution.

About EAI

EAI is a grassroots member organization initiated through cooperation between businesses, public, private and government organizations to address the global challenges of Europe's future competitiveness and link the European Research community with its counterparts around the globe. EAI reaches out to hundreds of thousands of individual subscribers on all continents and collaborates with an institutional member base including Fortune 500 companies, government organizations, and educational institutions, provide a free research and innovation platform.

Through its open free membership model EAI promotes a new research and innovation culture based on collaboration, connectivity and recognition of excellence by community.

More information about this series at http://www.springer.com/series/15427

Rashid Mehmood • Simon See • Iyad Katib
Imrich Chlamtac
Editors

Smart Infrastructure and Applications

Foundations for Smarter Cities and Societies

Springer

EAI
RESEARCH MEETS INNOVATION

Editors
Rashid Mehmood
High Performance Computing Center
King Abdulaziz University
Jeddah, Saudi Arabia

Simon See
Nvidia AI Technology Center
Singapore, Singapore

Iyad Katib
Faculty of Computing and Information Technology (FCIT)
King Abdulaziz University
Jeddah, Saudi Arabia

Imrich Chlamtac
European Alliance for Innovation
Gent, Belgium

ISSN 2522-8595 ISSN 2522-8609 (electronic)
EAI/Springer Innovations in Communication and Computing
ISBN 978-3-030-13704-5 ISBN 978-3-030-13705-2 (eBook)
https://doi.org/10.1007/978-3-030-13705-2

Preface

Infrastructure can mean several things and has been defined in several ways. It could refer to the fundamental systems and facilities that an organization, city, or country needs to function. Infrastructure could be provided by private, public, or public-private partnerships (PPPs) and could include roads, railways, bridges, airports, water, sewage, telephone, mobile and broadband equipment, manufacturing facilities, clinics and hospitals, schools and universities, and many more. Infrastructure could be hard or soft. Hard infrastructure is referred to as physical things such as ports, buildings, and electricity installations. Soft infrastructure refers to the institutions that facilitate organization and nations to function, and these include, for example, bodies, procedures, and programs for management, education, health, transport, law enforcement, and military. ICT infrastructures were considered separate from hard infrastructure in the past; however, with the increasing need for ICT penetration in societies, i.e., digital societies or economies, it is increasingly being considered part of the basic infrastructure.

Infrastructure needs to be understood in the context of the evolution of our societies, recent trends in urbanization, and the broader life. Smart cities provide the state-of-the-art approaches for urbanization, having evolved from the developments carried out under the umbrella of the knowledge-based economy and subsequently under the notion of digital economy and intelligent economy. Smart cities encompass all aspects of modern day life, transportation, healthcare, entertainment, work, businesses, social interactions, and governance. Smart cities exploit physical and digital infrastructure, as well as the intellectual and social capital, for urban and social development. Technically, smart cities are complex systems of systems that rely on converged and ubiquitous infrastructures. The smart city phenomenon is driven by several interdependent trends including a pressing need for environmental sustainability and peoples' increasing demands for personalization, mobility, and higher quality of life. The notion of smart cities can be extended to smart societies, i.e., digitally enabled, knowledge-based societies, aware of and working toward social, environmental, and economic sustainability.

Since knowledge and human and social capital are at the heart of the smart city and smart society developments, the role of education should extend beyond

the mainstream "education for employment" scope. It should extend to the notion of social and collaborative governance where the society collaborates to train each other in maintaining its knowledge, moral fiber, operations, good practice, resilience, and competitiveness and for bringing innovation and becoming a knowledge-based economy. The key to such efforts would be the creation of an ecosystem of digital infrastructures that are able to work together and enable dynamic real-time interactions between various smart city subsystems.

The word infrastructure was imported from French in English in 1887. The prefix "infra" in it means "below" which implied "structures" which mostly were underground such as water and sewage systems and tunnels. However, the meaning of infrastructure is continuing to be broadened, and the prefix "infra" could be taken as "foundational," i.e., "foundational structure," hard, soft, virtual, and digital, everything that we use today and we will use in the future, to support smart life. We define smart infrastructure as "*knowledge-based, collaborative, converged, ubiquitous, self-aware, adaptive, resilient, digitally-enabled, and self-governing foundational structure; comprising hard, soft, virtual, and digital facilities and systems, and intellectual and social capital; enabling social, environmental and economic sustainability; enabling innovation and competitiveness; facilitating personalization in all aspects of modern-day and future living, the aspects including transportation, healthcare, entertainment, work, businesses, social interactions, and governance; to meet societal, economic and other demands of organizations, cities or countries*." Smart infrastructure would include the Internet of Things (IoT) to monitor and actuate. High-performance computing (HPC), big data, artificial intelligence, cloud, fog, and edge computing will be needed to provide the necessary intelligence, storage, compute, and communication resources for the smart infrastructure.

We are delighted to introduce this book, which brings 26 chapters together on cutting-edge topics related to smart infrastructure and applications. Thirteen of these chapters are invited extended versions of papers from the proceedings of the first EAI (European Alliance for Innovation) Conference on Smart Societies, Infrastructure, Technologies and Applications (SCITA 2017) held at King Abdulaziz University (KAU), Jeddah, Saudi Arabia, on 27–29 November 2017 (see https://www.springer.com/la/book/9783319941790). The book is divided into two major themes and five parts. The first theme contains chapters where the focus is on the applications, in contrast to the second theme where the contributed chapters are mainly focused on infrastructure. Looking at the titles of the contributed chapters in this book, the distinction between the applications and infrastructure could be subtle in many cases because many applications eventually become part of the infrastructure and vice versa. The placement of the chapters in the two themes is based on the primary objectives and focus of the chapters, except in a couple of cases where the chapter placement is prioritized to keep the book structure.

Chapter 1 looks at enterprise systems and the role that they will play in the conceptualization and implementation of networked smart cities, particularly the information systems aspects of smart infrastructure. The "networked" aspect of smart cities is emphasized because smart cities will integrate its multiple subsystems

to create operational dynamicity and efficiency among other goals. The authors review a good number of conceptual definitions of smart cities to derive its system requirements. The technological foundations of smart cities and societies are reviewed along with many smart city applications from the literature. Partial least square regression is described as a method to model various interdisciplinary smart city constructs along with example applications from the literature. We have placed this chapter at the beginning of the book outside the five book parts because it provides foundational material on smart cities and infrastructures.

The applications theme of the book comprises 13 chapters divided into 3 parts. Part I includes seven chapters related to smart transportation. The first three chapters of this part are focused on detection of events or incidents using Twitter or inductive loops data. Chapter 2 proposes a methodology for analyzing traffic-related tweets in the Arabic language using SAP HANA. A technique is proposed for sentiment classification using lexicon-based approach to understand driver's feelings. The tweets are collected from Jeddah and Makkah cities (Saudi Arabia) in order to identify the most congested roads in the cities and to detect events such as accidents, roadworks, fire, and weather conditions. Chapter 3 is aimed at developing data management and analysis techniques for smart societies. It specifically uses big data, machine learning, and other platforms including Spark, MLlib, Tableau, and Google Maps Geocoding API, to study Twitter data for the detection and validation of spatiotemporal events in London. It empirically demonstrates that physical, virtual, and conceptual events can be detected automatically by analyzing data. It finds and locates congestion around London and the occurrence of multiple events including "London Notting Hill Carnival 2017" and their locations and times, without any prior knowledge of the events. Chapter 4 brings together transport big data, deep learning, in-memory computing, and GPU computing to predict traffic incidents on the road. Three different kinds of datasets are combined together to predict road traffic incidents. The three datasets include road traffic characteristics dataset, vehicle detector station (VDS), and incident data, acquired from the California Department of Transportation (Caltrans) Performance Measurement System (PeMS).

Chapter 5 provides a review and tutorial on a hybrid statistical machine learning method for big data road traffic modelling. The method is based on ARIMA (auto-regressive integrated moving average) and SVM (support vector machine). The authors use GPS road traffic data for prediction. Chapter 6 extends the authors' earlier work where they had developed a methodology to integrate supervised learning and decision fusion to enhance object classification accuracy in a driving environment. This chapter extends their earlier work and provides an in-depth performance comparison of deep learning and C5.0 decision tree classifier for object classification in driving environments using a bigger dataset. Chapter 7 uses road traffic data made publicly available by the UK Department for Transport and provides an extended analysis of a disaster management system that they have proposed in their earlier work. Chapter 8 proposes a system called Big Data Shortest Path Graph Computing (BDSPG) system for single-source shortest path computations of big data road network graphs using Apache Spark. They use the US

road network data, modelled as graphs, and calculate shortest paths between a set of large numbers of vertices in parallel on a supercomputer. Spark's parallelization behavior is analyzed by solving problems of varying graph sizes, various states of the USA (with over 58 million edges), and a varying number of shortest path queries reaching up to one million.

Part II includes three chapters related to smart healthcare. The use of DNA typing or profiling is rapidly growing in smart applications such as for the diagnosis of genetic diseases, paternity tests, and criminal identification. Chapter 9 provides an extended review of DNA profiling methods and tools with a particular focus on their computational performance and accuracy. The computational complexity of DNA typing increases significantly with the number of unknowns in the mixture. Faster interpretations of DNA mixtures with a large number of unknowns and higher accuracies are expected to open up new frontiers for this area. Chapter 10 proposes methods and architecture to improve cloud security for healthcare. Chapter 11 presents a review on the use of big data in healthcare supply chains with topics including big data, big data analytics, the role of big data in healthcare, supply chain management (SCM), healthcare supply chain management, and the role of Twitter data in SCM.

Part III includes three chapters on a mix of smart applications. Chapter 12 proposes a framework that uses mobile and cloud computing technologies to provide context-aware and portable recommendations for smart markets. The underlying algorithms and a prototype are developed to support automation, user intervention, and customization of users' preferences during the recommendation process. Chapter 13 uses association rule mining to gain insight into grades of a set of computer science students. Chapter 14 proposes SelecWeb, an automatic tool for selecting a Web framework based on a set of criteria and developer preferences. The authors develop the set of selection criteria using the analytic hierarchy process (AHP) and provide a detailed description and analysis of the tool including a case study for the Web framework selection.

The second theme of the book focuses on smart infrastructure with its 12 chapters, divided into 2 parts. Part IV presents a selection of nine chapters on big data, high-performance computing (HPC), and their convergence. The first four chapters are more focused on HPC. Chapter 15 investigates and presents the design and implementation of Hadoop clusters using ARM-based single-board computers (SBCs). The cost, energy consumption, and performance of the SBC clusters are discussed. Chapter 16 investigates the performance of parallel implementations of the Jacobi iterative method on Intel MIC Knights Corner (KNC) architecture looking at execution time, offloading time, and speedup. Chapter 17 reviews important performance characteristics of sparse matrix-vector (SpMV) computations on graphics processing unit (GPU) architectures along with various strategies to improve SpMV performance. Several well-known SpMV storage and solution schemes are discussed. SpMV multiplication is an essential building block for numerous scientific and engineering smart applications. Chapter 18 provides a review of performance analysis tools and techniques for HPC applications and

systems along with common HPC applications and a comparative analysis of HPC benchmarking suites.

The next four chapters of Part IV are focused on big data. Chapter 19 presents a review of the state-of-the-art tools and techniques for the processing of big data applications. In doing so, it critically analyzes their objectives, methodologies, and key approaches to address the challenges associated with big data. A detailed review and taxonomy of the research efforts of few core applications are provided. Although this chapter discusses applications, it is included in the infrastructure part due to its major focus on tools and technologies. Chapter 20 discusses opportunities, issues, and challenges of big data with the focus on the Hadoop platforms taking perspectives on data locality, load balancing, heterogeneity issues, scheduling issues, in-memory computation, multiple query optimizations, and I/O issues of big data. A taxonomy of big data opportunities and challenges is also proposed. Chapter 21 provides a review of the technologies related to software quality in emerging big data, IoT, and smart city environments. The roles of model checking and big data in software quality are discussed. Chapter 22 discusses the growing significance of open software and open data licenses in big data and smart infrastructures and proposes frameworks for the selection of open-source software and open data licenses. A review of notable open-source and open data licenses and the suitability of these licenses for various kinds of data and software is provided.

Chapter 23 concludes Part IV with a review of and a proposed architecture for big data and HPC convergence. The driving forces, challenges, and current and future trends associated with the integration of HPC and big data are identified. The programming models and frameworks of big data and HPC are reviewed, and their challenges in the exascale computing era are discussed.

Book Part V provides a selection of three chapters on IoT. Chapter 24 proposes a test execution platform based on the TTCN3 standard for dynamically adaptable IoT networks in smart cities. The platform considers both structural and behavioral adaptations and affords a platform-independent test system for isolating and executing runtime tests. Chapter 25 proposes a hierarchical clustered dynamic source routing (HCDSR) technique to improve fault tolerance and energy-efficient routing for wireless sensor networks (WSNs) in IoT environments. HCDSR is evaluated using simulations and compared with LEACH (low-energy adaptive clustering hierarchy) and DFTR (dynamic fault-tolerant routing) protocols. Chapter 26 proposes a model-based approach for testing security aspects of the Internet of Things for smart cities which consists of (a) modelling the system under investigation with an appropriate formalism, (b) deriving test suites from the obtained model, (c) applying some coverage criteria to select suitable tests, (d) executing the obtained tests, and (e) finally collecting verdicts and analyzing them in order to detect errors and repair them.

The book comprises research articles, and hence, it is aimed at early to advanced researchers. Some of the chapters are written in a tutorial manner and therefore can also be used for teaching purposes in universities.

We would like to acknowledge the support of many people who helped realize the publication of this book. The various committees of SCITA 2017 including the

TPC are acknowledged for their contributions in reviewing the papers, which are included in this book. Special thanks go to Eliška Vlčková, the managing editor at the European Alliance for Innovation (EAI), whose help and patience have been fundamental in bringing this book to publication.

This book is being introduced in an important time when so much is happening in ICT and smart infrastructure space. Many new smart districts and cities are being built around the world, while many exiting cities are evolving or transforming into smart cities. In Saudi Arabia, it is also a very high time with the recent announcement of its plans to build a smart city called NEOM supported by $500 billion from the Saudi government. This book will contribute to and shape these smart developments in Saudi Arabia and globally.

Jeddah, Saudi Arabia — Rashid Mehmood
Singapore — Simon See
Jeddah, Saudi Arabia — Iyad Katib
Gent, Belgium — Imrich Chlamtac

Contents

About the Editors

Rashid Mehmood is the Research Professor of Big Data Systems and the Director of Research, Training, and Consultancy at the High Performance Computing Center, King Abdulaziz University, Saudi Arabia. He has gained qualifications and academic work experience from universities in the UK including Cambridge and Oxford. Rashid has over 20 years of research experience in computational modelling and simulation systems coupled with his expertise in high-performance computing. His broad research aim is to develop multidisciplinary science and technology to enable a better quality of life and smart economy with a focus on real-time intelligence and dynamic system management. He has published over 150 research papers including 5 edited books. He has organized and chaired international conferences and workshops in his areas of expertise including EuropeComm 2009 and Nets4Cars 2010–2013. He has led and contributed to academia-industry collaborative projects funded by the Engineering and Physical Sciences Research Council (EPSRC), EU, UK regional funds, and Technology Strategy Board UK with the value over £50 million. He is a Founding Member of the Future Cities and Community Resilience (FCCR) Network. He is a Member of the Association for Computing Machinery (ACM) and The Optical Society (OSA), Senior Member of the Institute of Electrical and Electronics Engineers (IEEE), and former Vice-Chairman of IET Wales SW Network.

Simon See is currently the Solution Architecture and Engineering Director and Chief Solution Architect for Nvidia AI Technology Center. He is also a Professor and Chief Scientific Computing Officer in Shanghai Jiao Tong University. Professor See is also the Chief Scientific Computing Advisor for BGI (Beijing Genomics Institute, China) and has a position in Nanyang Technological University (Singapore) and King-Mong Kung University of Technology (Thailand). Professor See is currently involved in a number of smart city projects, especially in Singapore and China. His research interests are in the area of high-performance computing, big data, artificial intelligence, machine learning, computational science, applied mathematics, and simulation methodology. Professor See is also leading some of the AI initiatives in Asia Pacific. He has published over 200 papers in these areas and has

won various awards. Professor See is also a member of SIAM, IEEE, and IET. He is also a Committee Member of more than 50 conferences. Professor See graduated from the University of Salford (UK) with a Ph.D. in Electrical Engineering and Numerical Analysis in 1993. Prior to joining Nvidia, Prof. See worked for SGI, DSO National Lab. of Singapore, IBM, International Simulation Ltd (UK), Sun Microsystems, and Oracle. He is also providing consultancy to a number of national research and supercomputing centers.

Iyad Katib is an Associate Professor with the Computer Science Department and the current Vice Dean and the College Council Secretary of the Faculty of Computing and Information Technology (FCIT) in King Abdulaziz University (KAU). He is also the Director of KAU High Performance Computing Center. Iyad received his Ph.D. and M.S. in Computer Science from the University of Missouri-Kansas City in 2011 and 2004, respectively. He received his B.S. in Statistics/Computer Science from King Abdulaziz University in 1999. His current research interest is on computer networking and high-performance computing.

Imrich Chlamtac is the President of **EAI**, the European Alliance for Innovation, where he pioneered EAI as a global initiative for promoting growth of ICT-based economy and digital society. Based on community cooperation principles, EAI supports research and innovation with events, publications, and its innovation platform through the cooperation of over hundred and fifty thousand subscribers worldwide. Dr. Chlamtac is also the Founding President of **CREATE-NET** and Bruno Kessler Professor at the University of Trento, top ranked ICT institutions in Italy.

Prior to coming to Europe Dr. Chlamtac served as Associate Provost for Research and Distinguished Chair in Telecommunications at the University of Texas in Dallas, the number one "young university" in the USA in the Times of Higher Education ranking 2017. Prior to that he was a Professor at Boston University, University of Massachusetts, and Technion.

Dr. Chlamtac holds multiple academic and honorary appointments including at the University of Trento, the Tel Aviv University, the Beijing University of Posts and Telecommunications, and the Budapest University of Technology and Economics.

Dr. Chlamtac scientific recognitions include IEEE and ACM Fellowship, the ACM Award for Outstanding Contributions to Research on Mobility, the IEEE Award for Outstanding Technical Contributions to Wireless Personal Communications, New Talents in Simulation of SCS, Fulbright Scholarship, and IEEE Distinguished Lecturer. He was listed in ISIHighlyCited.Com among the 250 most cited computer science researchers worldwide. Dr. Chlamtac is also included in the list of Notable People from the University of Minnesota where he received his PhD and holds an Honorary Citizenship in Slovakia where he was born.

Dr. Chlamtac published over 400 refereed articles and multiple books. He is the coauthor of the first textbook on *Local Networks* (Lexington Books 1980) and IEEE Network Editor's choice and Amazon.com engineering books best seller *Wireless and Mobile Network Architectures* (John Wiley & Sons 2000).

As part of his contribution to the research community, Dr. Chlamtac founded the ACM SigMobile, serves as Editor in Chief of the Springer WINET and MONET journals, and established ACM Mobicom and other leading conferences. As the original architect and current President of EAI (http://www.eai.eu/) he is leading the scientific development of one of the largest research communities in Information Sciences and their impact on society.

Dr. Chlamtac is a co-founder and past President of CONSIP Ltd, the first network emulator company, and of BCN Ltd, currently KFKI Ltd (http://www.kfkizrt.hu/), one of the largest system integrator companies in Central Europe.

Chapter 1
Enterprise Systems for Networked Smart Cities

Naim Ahmad and Rashid Mehmood

1.1 Introduction

The twenty-first century is witnessing unfathomable pace of change. Almost every decade some cutting-edge innovation transforms the way people are going to live in the next decade. This opens at one hand challenges to cope up with and on the other hand opportunities to live in the better world. One recent invention is the concept of smart city. A city that functions as a device and yet pleases the hearts and minds of its habitants. A city where in technology is part and parcel of everyday life but invisibly blended with nature and acts more like a natural entity. The concept sounds like utopia in the context that technology has perfectly been harnessed as peer to human in achieving the perfection in society.

The field of smart city is highly interdisciplinary and requires coordinated efforts from all the stakeholders such as city administrators and planners, government, academia, industry, and professionals. All the stakeholders must present unified and consistent vision of the smart city. Integration of physical, institutional, and digital aspects [1] is essential. Advancements in information and communication technology (ICT) have made it possible to start the journey of smart city.

Today there exists enough hardware and software frameworks to convert a city into a digital device. Internet of Things (IoT) and cloud computing together will provide the autonomous behavior to the real-world entities. Whereas, big data is capable to handle gigantic data streams generated every day. Enterprise

N. Ahmad
Department of Information Systems, King Khalid University, Abha, Saudi Arabia
e-mail: nagqadir@kku.edu.sa

R. Mehmood (✉)
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: RMehmood@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_1

systems technologies and frameworks are well suited to define, develop, and implement smart city systems. And web services and service-oriented architecture is ideal platform for the development of smart city systems. Further, semantic web technologies will help in achieving the autonomous behavior of the systems.

To further the cause of smart city, researchers must study and develop interrelationship models between different constructs. Partial least square regression, a structural equation modeling technique will be helpful in establishing the causal links between different constructs. This chapter sheds some light on the concept of smart city (Sect. 1.2), application of enterprise systems to develop city level integrated info-structure (Sect. 1.3), latest information technology innovation to build foundations (Sect. 1.4) and structural equation modeling to establish the interrelationship between multidisciplinary latent variables (Sect. 1.5). Conclusions are drawn with future research directions (Sect. 1.6).

1.2 The Concept of Smart City

The concept of smart cities originates from the excellence of multitudes of disciplines such as information technology, sustainability, architecture and urban planning, social and economic development to mention some. There are numerous definitions available in the literature on the smart city focusing on different aspects. Some of them are given below in the chronological order.

Hall et al. [2] presented vision of smart city as integration of science and technology through information system, and new relationships between government, city managers, business academia and the research community. And gave working definition as "A city that monitors and integrates conditions of all of its critical infrastructures, including roads, bridges, tunnels, rails, subways, airports, seaports, communications, water, power, even major buildings, can better optimize its resources, plan its preventive maintenance activities, and monitor security aspects while maximizing services to its citizens." He moreover predicted that "smart cities vision, systems and structures will monitor their own conditions and carry out self-repair as needed."

Giffinger [3] defined smart city as "A city well performing in a forward-looking way in these six characteristics (smart economy, smart people, smart governance, smart mobility, smart environment, and smart living), built on the smart combination of endowments and activities of self-decisive, independent and aware citizens." He identified thirty-three factors and 1–4 indicators to measure the six characteristics of the state of the smart city and ranked 70 mid-sized European cities. He stressed on reporting of metrics along with ranking to make the whole exercise more actionable.

Hollands [4] didn't give the formal definition but pointed that literature on smart city talks about "utilization of networked infrastructure to improve economic and political efficiency and enable social, cultural, and urban development," where the term infrastructure indicates business services, housing, leisure and lifestyle services, and ICTs (mobile and fixed phones, satellite TVs, computer networks, e-commerce, and internet services). He gives a discourse on how the overemphasis

on technology can develop false smart cities which can further aggravate the problems such as gentrification and power imbalance. Therefore, in his view smart city initiative should start from the people and ensure the balance of power between privileged class and ordinary people.

Washburn et al. [5] mention the smart city definition of Forrester as "The use of Smart Computing technologies to make the critical infrastructure components and services of a city—which include city administration, education, healthcare, public safety, real estate, transportation, and utilities—more intelligent, interconnected, and efficient" and smart computing as "A new generation of integrated hardware, software, and network technologies that provide IT systems with real-time awareness of the real world and advanced analytics to help people make more intelligent decisions about alternatives and actions that will optimize business processes and business balance sheet results." They assert that the CIOs are the primary enablers of smart cities. They have defined the roles of CIOs such as city level CIO, critical public infrastructure and services related CIOs, and CIOs related with entities that act as consumer of critical public infrastructure and services in planning, implementing and delivering the smart city vision.

Harrison et al. [6] have defined the smart city from the perspective of information technology as "connecting the physical infrastructure, the IT infrastructure, the social infrastructure, and the business infrastructure to leverage the collective intelligence of the city." They have presented the IT infrastructure that will improve the operational efficiency of the city and quality of life. They proposed the service-oriented architecture (SOA) model and mentioned its shortcomings as well such as it can handle only up to 1000 events per second.

Chen [7] defined the concept as "Smart cities will take advantage of communications and sensor capabilities sewn into the cities' infrastructures to optimize electrical, transportation, and other logistical operations supporting daily life, thereby improving the quality of life for everyone." The focus of his note challenged the capability of existing network infrastructure that is based on TCP/IP, the protocol that utilizes the stateless routers and trusts the hosts. He emphasizes on the new inherent capabilities in the network such as fast self-healing, sender authentication, and per-hop packet accounting for packet trace back to support smart city critical infrastructure.

Caragliu et al. [8] identified the characteristics of smart cities through literature review as: utilization of networked infrastructure, business-led urban development, social inclusion, high-tech and creative industries, social and relational capital, and social and environmental sustainability. Further they gave the operational definition as "city is smart when investments in human and social capital and traditional (transport) and modern (ICT) communication infrastructure fuel sustainable economic growth and a high quality of life, with a wise management of natural resources, through participatory governance." Based on the analysis of Urban Audit data for the years 2003–2006 (more than 250 indicators for around 260 cities of EU27 nations) they found the positive correlation between urban wealth (measured using PPP), and the presence of vast number of creative professionals, a high score in multimodal accessibility indicators, the quality of urban transportation networks, the diffusion of ICTs, and the quality of human capital.

Velosa et al. [9] defines the smart city as "A smart city is based on intelligent exchanges of information that flow between its many different subsystems. This flow of information is analyzed and translated into citizen and commercial services. The city will act on this information flow to make its wider ecosystem more resource efficient and sustainable. The information exchange is based on a smart governance operating framework designed to make cities sustainable."

Komninos [1] brings the concept of spatial intelligence to describe the phenomenon of smart cities. The spatial intelligence materializes from the agglomeration and integration of intelligence of intellectual capital, social capital and infrastructural capital. And emergence and utilizations of this spatial intelligence to solve diverse problems makes a city smart. He also poised that sustainable knowledge economy and internet are the primary drivers for the smart cities. Thereafter from cases of Bletchley Park, Cyberport Hong Kong, and Amsterdam Smart City, he brings out three types of spatial intelligence such as orchestration intelligence (integration along community-based workflows), amplification intelligence (integration of skills, digital tools, and city infrastructures), and instrumentation intelligence (integration of city infrastructure, activity data flows, measurement devices, and predictive modeling). These spatial intelligences can be adopted in isolation or in combination to develop smart cites. He also mentions that the smart cities should address the challenges of competitiveness through knowledge economy, efficient employment market to reduce poverty and environmental sustainability as pointed out in the report of European Commission, 2008. Finally, he points out that research should focus on to develop architectures of integration between physical, institutional and digital aspects of cities to have higher special intelligence that will lead to real smart city.

Nam and Pardo [10] have proposed the strategies to make a smart city. They gave the working definition of it as "A smart city infuses information into its physical infrastructure to improve conveniences, facilitate mobility, add efficiencies, conserve energy, improve the quality of air and water, identify problems and fix them quickly, recover rapidly from disasters, collect data to make better decisions, deploy resources effectively, and share data to enable collaboration across entities and domains." Then they identified three core factors such as technology, human and institution critical to achieve the vision of smart city. Technology factor helps in integrating the technologies, systems, infrastructures, services, and capabilities into an organic network. Human factor should facilitate learning more so the social learning that can bridge the gap of digital divide. Institution factor will provide dynamic governance to connect citizens, communities, and business in real time to spark growth, innovation, and progress in assistance with the pivotal and visionary role of leadership.

Thite [11] puts forth the point that smart city needs smart people and using the economic geography theory illustrates on the factors that influence knowledge worker in choosing a place to live and work. From this perspective he posited that "Smart city aims at nurturing a creative economy through investment in quality of life which in turn attracts knowledge workers to live and work in smart cities." The New Economic Geography relating to "economies of scale and spatial

development" describes the agglomeration of firms and workers. On the other hand, New Neoclassical Urban Economics relating to "optimal preference- satisfying behaviors" describes the joining of more firms and individuals.

Rios [12] defined a smart city as "A city that gives inspiration, shares culture, knowledge, and life, a city that motivates its inhabitants to create and flourish in their own lives." His emphasis is on to create empowered spaces that help diverse people and culture to achieve identity and generate ideas. And the architectural design of city should cause motivation to create great places, inspiration (connectivity) to become entrepreneur, innovation to reinvent themselves, and identity to become leader.

Thuzar [13] defined the concept as "Smart cities are cities that have a high quality of life; those that pursue sustainable economic development through investments in human and social capital, and traditional and modern communications infrastructure (transport and information communication technology); and manage natural resources through participatory policies. Smart cities should also be sustainable, converging economic, social, and environmental goals." He emphasized the correlation of smart city with the livability and entities to promote prosperity, equity, and sustainability.

Barrionuevo et al. [14] proposed "Being a smart city means using all available technology and resources in an intelligent and coordinated manner to develop urban centers that are at once integrated, habitable, and sustainable." They identified five types of capital such as economic, human, social, environmental, and institutional that can be nurtured through innovation, social cohesion, sustainability, and connectivity. They also presented the step by step process along with timeline to build a smart city such as diagnose the situation (2–5 months), develop a strategic plan (5–12 months) and take action (2–10 years). Further city must assess the levers of change such as strategic and scenario planning; collaboration and communication; public–private partnerships; funding strategies; capacity management; and technological infrastructure in the context of its competitive situation and strategic position.

Cretu [15] identified two main streams of research ideas with respect to smart cities: "(1) smart cities should do everything related to governance and economy using new thinking paradigms and (2) smart cities are all about networks of sensors, smart devices, real-time data, and ICT integration in every aspect of human life." Exploring further the second dimension he proposes three axioms regarding a smart city that it has well-designed ICT infrastructure, transforms real-time data into meaningful information, and allows inhabitants to predefined automated action in response to events. He proposed the system perspective for smart city as "Event-driven Smart City (EdSC) as a system (software platform) representing an internet-aware digital living, environment where people, software services, sensors and smart devices interact by means of events and listeners." The basic ingredients of this platform are ability to convert signal to event; knowledge sharing; action definition, storage and event-condition-action (ECA) relationship; and ability to convert action into signals. The Smart Community Space (SCS) is the central architectural component of EdSC that implements the partial view

over the world. The SCS is built on subset of common interest such as family, school, and physical area in order to sustain complexity when it comes to real-time data processing and complex event processing (CEP). EdSC requires event driven architecture (EDA) and semantic event processing based on ECA. That will use different ontologies and domain knowledge to remove ambiguity with events. And the modular event ontologies can also encompass different languages in addition to different ontologies. Semantic web technologies such as RDF (Resource Description Framework), LOD (Linked Open Data) Cloud, OWL (Web Ontology Language), SWRL (Semantic Web Rule Language), RIF (Rule Interchange Format), SBVR (Semantics of Business Vocabulary and Business Rules), and SPARQL (RDF Query Language) are useful in implementing the EdSC. RDF can be used to represent event and ontologies in LOD Cloud to attach semantic to identify the meaning of signals in EdSC platform. OWL can be used in signal-to-event transformer, CEP algorithms, and identification of listeners for particular events. SWRL and RIF are apt for defining listeners and SBVR can provide the natural language extension. And SPARQL can provide the advanced support for querying the right listeners for the particular event.

Kourtit and Nijkamp [16] define the concept as "Smart cities are the result of knowledge-intensive and creative strategies aiming at enhancing the socio-economic, ecological, logistic and competitive performance of cities. Such smart cities are based on a promising mix of human capital (e.g. skilled labor force), infrastructural capital (e.g. high-tech communication facilities), social capital (e.g. intense and open network linkages) and entrepreneurial capital (e.g. creative and risk-taking business activities)." They also assert that the productivity gain in smart city should offset the rise in the local problems. These places attract and retain variety of creative people and offer innovative and sustainable solution. And these places exploit the agglomeration advantages to its maximum.

Kourtit et al. [17] posit the concept as "Smart cities have high productivity as they have a relatively high share of highly educated people, knowledge-intensive jobs, output-oriented planning systems, creative activities and sustainability-oriented initiatives." They identified that the knowledge orientation and cultural diversity are the key factors in present era of smart cities. The physical, geographical, and social proximities amplify the agglomeration effects to facilitate knowledge processes. ICT can help in codified knowledge but fails in tacit and contextual knowledge. Therefore, the proximities and agglomeration phenomenon remain important.

Lazaroiu and Roscia [18] define the concept of smart city as "A community of average technology size, interconnected and sustainable, comfortable, attractive and secure." They proposed the smart city assessment model having 18 indicators devised with the help of fuzzy logic.

Lombardi et al. [19] define the concepts as "The application of information and communications technology (ICT) with their effects on human capital/education, social and relational capital, and environmental issues is often indicated by the notion of smart city." They have proposed a smart city assessment model with the help of ANP (Analytical Network Process) technique. They developed their model on the four helices principal such as Civil Society, University, Industry, and

Government. Further, they identified the civic involvement along with social and cultural capital endowments, shapes the relationship between the triple helices such as university, industry and government. And the interplay between all four helices including the civic society determines the progress of smart city. They modeled the five clusters (Smart Government, Smart Economy, Smart Human Capital, Smart Living, and Smart Environment), four helices (Government, University, Industry, and Civil Society), and four policy visions (Connected City, Entrepreneurial City, Livable City, and Pioneer City) along with the indicators. The result was that all the clusters are trying to pursue the policy strategy of entrepreneurial city.

Bakici et al. [20] in the context of Barcelona city posited "Smart city as a high-tech intensive and advanced city that connects people, information and city elements using new technologies in order to create a sustainable, greener city, competitive and innovative commerce, and an increased life quality with a straightforward administration and good maintenance system." Achieving competitiveness of the Barcelona has been the foremost factor to transform it into a leading smart city in Europe. The city has sagaciously used the advanced ICT technology to develop infrastructures such as Open Data, Living Labs, and Smart District. to foster the dynamism of triple helix (faculty, company and citizen interaction) to boost creativity, knowledge, and innovation and hence supports the conceptual model of Barcelona for smart city, knowledge economy and knowledge society.

Zygiaris [21] illustrates the concept as "A smart city is understood as a certain intellectual ability that addresses several innovative socio-technical and socio-economic aspects of growth. These aspects lead to smart city conceptions as "green" referring to urban infrastructure for environment protection and reduction of CO_2 emission, "interconnected" related to revolution of broadband economy, "intelligent" declaring the capacity to produce added value information from the processing of city's real-time data from sensors and activators, whereas the terms "innovating", "knowledge" cities interchangeably refer to the city's ability to raise innovation based on knowledgeable and creative human capital." He proposed six layers for smart city planning in addition to The City Layer as Layer 0. These six layers are: The Green City Layer, The Interconnection Layer, The Instrumentation Layer, The Open Integration Layer, The Application Layer, and The Innovation Layer. The City Layer (Layer 0) refers to the recognition of city's identity and planning of urban infrastructure in compliance with the smart city priorities and socially inclusive vision. The Green City Layer (Layer 1) refers to the environment friendly development such as CO_2 footprint reduction, alternative energy options, green transport management, and green building specification. The Interconnection Layer (Layer 2) refers to the holistic broadband connectivity to bring all communities online. The Instrumentation Layer (Layer 3) refers to real-time event aggregation from the range of sensors and actuators through Internet of Things framework. The Open Integration Layer (Layer 4) refers to the development of Urban Operating System to share the data, content and services. This platform is designed and implemented on technologies such as Semantic Web, Linked Open Data, Visualization of APIs, Internet of Trust and Cloud Computing. The Application Layer (Layer 5) refers to the development of intelligent application in

different domains such as e-traffic, e-government, e-democracy, and smart energy grids etc. The Innovation Layer (Layer 6) refers to the community wide utilization of urban innovation ecosystem of a smart city to develop new business models to navigate the city on the sustainable growth. Author also emphasizes upon the development of processes-oriented assessment model for smart city success rather than simple benchmarking indexes.

Marsal-Llacuna et al. [22] define the concept as "Smart Cities initiatives try to improve urban performance by using data, information and information technologies (IT) to provide more efficient services to citizens, to monitor and optimize existing infrastructure, to increase collaboration among different economic actors, and to encourage innovative business models in both the private and public sectors." And they also gave more technical definition of it as "cities wishing to become smart must be equipped with a "brain" (software) supplied with lots of real-time information (data collected from sensors) enabling them to take more sustainable, efficient and citizen-centric decisions, smoothly transforming decisions into actions by means of technological solutions." He emphasized on the normalization process of indicators and development of summarization index, essential to compare the smartness of the cities in addition to their rankings. And real-time data is more important than the statistical data for which ubiquitous computing, remote sensing imaginary, smart meters, etc. should be utilized.

Albino et al. [23] identified four common characteristics for the smart city "a city's networked infrastructure that enables political efficiency and social and cultural development; an emphasis on business-led urban development and creative activities for the promotion of urban growth; social inclusion of various urban residents and social capital in urban development; the natural environment as a strategic component for the future." Author also proposes that the assessment of smart city initiatives should be tailored to its unique characteristics and vision.

Mehmood et al. [24] define, "smart cities provide the state of the art approaches for urbanisation, having evolved from the developments carried out under the umbrella of knowledge-based economy, and subsequently under the notion of digital economy and intelligent economy." Smart society is an extension of the smart cities concept, "a digitally-enabled, knowledge-based society, aware of and working towards social, environmental and economic sustainability" [24]. Muhammed et al. [25] note that smart cities are considered a major driver for the transformation of many industries due to the fact that smart cities are driven by, or involve, integration of multiple city systems, such as transport, healthcare, and operations research, with the aim to provide its citizens a high quality of life.

To identify the frequent dominating themes of smart city, the word cloud has been generated from these definitions (see Fig. 1.1). Looking at the word cloud created from the definitions of smart cities three major areas can be identified, first is the enablement through advanced information and communication technology (ICT) infrastructure. Second is the achievement of highest standards of needs of a society or community such as knowledge, economy and life. And the third deals with the operationalization, meaning doing so in resource efficient way or sustainable way. These challenges or requirements can define the vision for the smart city system as follows.

Fig. 1.1 Word cloud generated from the smart city definitions

"Smart City System is a convergence of integrated systems supporting city processes reengineered on sustainability principles and utilizing state of the art technology to advance frontiers of knowledge, economy and life in a society." In the organizational context, these types of integrated systems have been existing since 1990s. They were labeled by different names such as Enterprise Systems (ES) or Enterprise Resource Planning (ERP). Also known with the application-specific modules such as customer relationship management (CRM) or supply chain management (SCM). Following section illustrates upon this innovative technology.

1.3 Enterprise Systems: Technology and Evolution

Enterprises face stiff competition and operate at times on raiser thin margins. Sustaining the business requires innovation and proactive approach in anticipation of future issues and problems. It is mandatory to maintain optimal productivity, excellent customer service and reduced cycle times. Enterprises have been applying ICT to gain competitive advantages since 1950s. This section gives the introduction of information technology in enterprises, technology of enterprise systems and its architecture.

1.3.1 Information Technology in Organizations

Investments in IT are ever increasing and none of the organization wants to miss this band wagon of IT. A sizable portion of 4.2% of annual revenue is spent on IT [26]. On an average organizations' 50 percent capital expenditure budget is utilized by IT that has steadily increased every decade from less than 5% in the year 1965 [27].

The core functions of the information technology are data storage, data transport, and data processing. The cost to carry out these functions is decreasing as hypothesized by Moore's Law "Every two years the number of transistors on integrated circuit doubles." The information technology can be thought of a bundle of shared services to cater to the need of communication and foundation for implementing business applications [28]. IT infrastructure is defined as a set of shared IT resources which is a foundation for both communication across the organization and the implementation of present/future business applications [29]. IT infrastructure is composed of two components, Technical consisting of hardware, software, network, telecommunications, applications, and tangible IT resources, and Human referring to knowledge and skill required to orchestrate the technical components [29].

Weill et al. [30] collected data belonging to the period of 1990–2001. They gathered the data for 180 business initiatives from 118 businesses in 89 enterprises. The enterprises selected were top three in their industry. After analysis of the data collected, they proposed that the IT infrastructure has to be thought of in terms of services since the agreement level can be made stable whereas underlying technology is more dynamic. These infrastructural services are deployed at multiple level enterprise wide or business unit level. Where to place a capability is a strategic decision pursued by the concerned organization. For instance, keeping a single point of customer contact requires capability to be developed at the enterprise wide scale so that the data can be shared and facilitate cross selling.

They further identified 70 IT infrastructural services and grouped them into 10 capability clusters. Six layers were defined as the physical layer of IT infrastructure capability, namely Channel Management, Security and Risk, Communications, Data Management, Application Infrastructure, and IT Facilities Management. And reaming four clusters IT Management, IT Architecture and Standards, IT Education, and IT Research and Development were considered as management-oriented IT infrastructure capability. Applications like enterprise systems such as Enterprise Resource Planning (ERP), Supply Chain Management (SCM), and Customer Relationship Management (CRM) fall under the cluster of application infrastructure.

Technical components of IT are readily available and more or less have become commodity input to the whole infrastructure. The distinction lies in skills and abilities to utilize them to gain strategic advantage. Whereas some authors like [27] have gone to the extent that IT doesn't add any strategic value and whole of IT has become an infrastructural technology similar to railroads, telegraph, electricity, etc.

1.3.2 About Enterprise Systems

Most organizations have, broadly speaking, following divisions procurement, production and operations inventory management, finance, marketing, and sales and distribution. Each department has got its own processes and procedures. Information and control from one department to another department need to be coordinated in order to execute a business. Traditionally these departments used to work in a very fragmented fashion. That often resulted in the creation of information silos.

Every department used to tackle its own IT initiatives often in isolation with the overall strategic direction of the organization. At times same software was implemented in multiple business units incurring unnecessary cost. As we know software is nothing, but a set of complex code written in a programming language. Due to infinite scenarios possible with any simple software, testing the software is a Herculean task. As time passed by these isolated software could not cope up with the important parameters of IT like scalability, portability, robustness, and integration.

The innovations in the information technology have led to the creation of a perfect network of information interchange that allows the removal of all the hassles in information sharing. Moreover, software packages have been developed that can ideally suit to any organization's processes. These packages are known as enterprise systems that are set of applications that interconnect the different processes and procedures of the organization. They utilize a central database for all the data needs [31].

1.3.3 Evolution of Enterprise Systems

Enterprise systems are also known as enterprise applications or business applications. These systems allow the seamless integration of information flow in the organizations internally and externally. The current ES have a long evolutionary history with them. The promoters of ES have been broadening the canvass of integration from financial and accounting, production and manufacturing, marketing and sales, logistics and distribution, human resource to strategic functions [31]. The offerings are increasing day by day along with the complexity and failures of implementations [32]. Accordingly, researchers and practitioner have been labeling these systems as they are growing.

Inventory management and control processing software were famous in the fifties and sixties. These systems were developed on mainframe platform using third-generation programming languages such as Fortran and Cobol. The focus of these systems was on managing and tracking inventory effectively and efficiently by automating inventory management and production schedules.

The next in line were Manufacturing Resource Planning (MRP) software in the seventies. This software was developed on the same technological paradigm as the previous one. The focus of these systems got enlarged to include sales and marketing

by linking the planning of product or parts requirements to the master production schedule. The next version of manufacturing resource planning was termed as MRP II. They were developed on the mainframe legacy platform using fourth-generation database software and manufacturing applications. The focus of these systems was further refined to manufacturing strategy and quality control, and included the support for designing production supply chain processes.

In the nineties, the most comprehensive software packages, Enterprise Resource Planning (ERP) came into existence. These systems had originated from MRP and MRP II [33]. They were developed on multiple platform mainframe and client-server using fourth-generation database software and packages software application. The focus of these systems was application integration and customer service by automating and optimizing all the processes sales and distribution, finance and accounts, human resource, procurement, etc.

At one end promoters were trying to integrate information flow internally and ended up in developing ERP packages. On the other hand, some software vendors were working on to integrate the external information flow from the customer and supplier side and developed CRM and SCM software packages. The ERP vendors integrated the functionalities of SCM and CRM in their packages [34]. Hence researchers and practitioners have started using the term ERP II [35]. These systems are developed on web-based client-server platform and integrated with fifth-generation applications like SCM, CRM, and SFA etc. The focus of these systems is agility and customer-centric global environment by e-enablement. Table 1.1 describes the timeline, system, and platform they utilized.

1.3.4 Definitions of Enterprise Systems

The term ERP in the press was first used in 1992 by Lopes and in the year 1996 Davenport introduced it to IS community at AMCIS ‘96 and called these packages metapackages [36]. ERP systems are said to have packaged processes for best business practices as business blueprint that can guide the organization for product engineering, evaluation and analysis, and implementation [33].

The term enterprise systems was in use since 1980s to refer to any enterprise wide integrated systems [37]. Davenport [38] used the same term enterprise systems instead of ERP in his famous article “Putting the enterprise into the enterprise system.” Thereafter academic fraternity prefers to use the term enterprise systems and includes many other enterprise wide integrated systems such as SCM, CRM, and PLM. Few of the definitions of enterprise systems are given in the following paragraphs.

“An, integrated, multi-dimensional system for all functions, based on a business model for planning, control, and global (resource) optimization for the entire supply chain, by using the state of the art IS/IT technology that supplies value added services to all internal and external parties” [39].

Table 1.1 Timeline of evolution of enterprise systems [40]

Timeline	System	Platform	Description
1960s	Inventory management and control	Mainframe legacy using third-generation software (e.g., Cobol, Fortran)	These systems were designed to manage and track inventory efficiently and help the plant supervisors on purchase orders, alerts, targets, providing replenishment techniques and options, inventory reconciliation, and inventory report.
1970s	Material requirement planning (MRP)	Mainframe legacy using third-generation software (e.g., Cobol, Fortran)	With the focus on sales and marketing, these systems were job shop scheduling processes. MRP generated schedule for production planning, operations control, and inventory management.
1980s	Material requirement planning II (MRP II)	Mainframe legacy using fourth-generation database software and manufacturing applications	With a focus on manufacturing strategy and quality control, these systems were designed for helping production managers in designing production chain processes—from production planning, parts purchasing, inventory control, and overhead cost management to product distribution.
1990s	Enterprise resource planning (ERP)	Mainframe or client server using fourth-generation database software and package software application to support most organizational functions	With a focus on application integration and customer service, these systems were designed for improving the performance of internal business processes across the complete value chain of the organization. They integrate both primary business activities like product planning, purchasing, logistics control, distribution fulfillment and sales; additionally, they integrate secondary or support activities like marketing, finance, accounting, and human resource
2000s	Extended ERP or ERP II	Client-server using the web platform, open source and integrated with fifth-generation applications like SCM, CRM, SFA. Also available on software as a service (SaaS) environment	With a focus on agility and customer centric global environment, these systems extended the first-generation ERP into inter-organizational systems ready for e-business operations. They provide anywhere anytime access to resources of the organization and their partners; additionally, they integrate with newer external business modules such as supply chain management, customer relationship management, sales force automation (SFA), and advanced planning and scheduling (APS)

American Production and Inventory Control Society (APICS) defines ERP as "a method for the effective planning and controlling of all the resources needed to take, make, ship and account for customer orders in a manufacturing, distribution or service company."

"Enterprise Systems (ES) are typically comprehensive, complex, customizable integrated application software that support core business processes and main administrative areas of enterprises in different industries" [37].

"ERP is a standardized software packaged designed to integrate the internal value chain of an enterprise. An ERP system is based on an integrated database and consists of several modules aimed at specific business functions" [35].

In the sustainable future city context enterprise systems have been defined as "ES that supports the integration, management, and regulatory compliance of environmental, social and economic sustainability, in addition to providing support for all internal and external business processes and organizational information needs to enhance firm's performance, resilience and sustainability" [41].

1.3.5 Business Process

Enterprise systems break through the traditional functional boundaries and try to automate a business process such as order fulfillment. Process view is very critical while implementing enterprise systems. The process has been defined as "a set of logically related tasks performed to achieve a defined business outcome" [42]. A process is "a structured, measured set of activities designed to produce a specified output for a particular customer or market. It implies a strong emphasis on how work is done within an organization" [38]. There are numerous modeling methods for business process modeling. One such IDEF3 method captures the process where domain expert can define a scenario as a set of ordered events along with the participating objects [43].

1.3.6 Components of Enterprise Systems

The first wave of success of ES was more focused on the internal processes of the organization and the packages were termed as ERP. The success of ERPs provided impetus for the ES vendors to venture in the other dimensions. And the major thrust was given to logistics and customer relationship management since both of these processes have to be optimized for success of any enterprise. Therefore, these emerging business needs led to the concept of ERP II [35].

The core components deal with the distributed central database and application framework such as .NET or J2EE (Table 1.2). The central components consist of ERP as a transaction processing system with all the traditional sub-components sales and distribution, finance and accounts, human resource, procurement, etc. In addition, it also contains business process management tools to design, execute, and evaluate business processes [35].

Table 1.2 The four layers in ERP II [35]

Layer	Components	
Foundation	Core	Integrated database (DB) Application framework (AF)
Process	Central	Enterprise resource planning (ERP) Business process management (BPM)
Analytical	Corporate	Supply chain management (SCM) Customer relationship management (CRM) Supplier relationship management (SRM) Product lifecycle management (PLM) Employee lifecycle management (ELM) Corporate performance management (CPM)
Portal	Collaborative	Business-to-consumer (B2C) Business-to-business (B2B) Business-to-employee (B2E) Enterprise application integration (EAI)

The third layer is more analytical in nature and provides the tools for management to answer the challenges in a timely fashion. Supply chain management (SCM) helps in the planning and production of goods. On the other hand, SRM, CRM, ELM, and PLM help in maintaining the lifecycle of supplier, customer, employee, and product, respectively. Corporate performance management (CPM) provides the indices and matrices to see the overall performance of the organization.

Collaborative layer takes the business online and provides the window to the external and internal world without any bias or control. It provides a portal for customers (B2C), suppliers (B2B), and employees (B2E) to transact with the organization from the internet platform using a simple client like a web browser. In addition, it also provides a mechanism for the integration of third-party systems via EAI.

1.3.7 The Architecture of Enterprise Systems

Enterprise systems have three distinct characteristics in their architecture data dictionary, middleware and repository [33]. Data dictionary contains thousands of domains along with their fields that can be used in all functions or entire value chain of the organization. Middleware can allow users to set up software routines and databases at different location that can route data intelligently. Repository at the base acts as a business framework containing semantics of business processes, business objects and organizational model [44]. It consists of complete information of application including metainformation on models, business objects, and technical programming objects [33].

1.4 Technological Foundations of Smart Cities

This section will discuss the latest advancements in the area of ICT that can be leveraged to lay the foundation of the smart city systems. As the humongous size and complexity of city will require resilient ICT services, service oriented architecture is an ideal way to develop services as opposed to monolithic huge software. Similarly, the computing resources should not be a constraint, rather expandable cloud computing model should be utilized. Sensors and actuators are going to be an indispensable part of smart city and should be integrated through internet of things. To make sense out of daily trillions of bytes of new data, big data technology is essential. Likewise, in highly mechanized world concepts of semantic web should be implemented to increase the machine-to-machine interactions. Finally, this section concludes with possible modules of the smart city systems.

1.4.1 Service-Based Distributed Computing

Service-based computing is one of the fundamental technologies driving smart cities developments. Cloud computing can be considered as a stage in the realization of service-based computing. Cloud computing paradigm offers virtually unlimited computing. The National Institute of Standards and Technology (NIST) defines it is defined as "a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction" [45]. And its "capability to serve on-demand, share and instant commissioning and de-commissioning of configurable computing resources causes it to be resilient, sustainable and near-utility computing" [46].

Cloud computing is offered majorly through Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). And deployment could be private, public, community, or hybrid cloud. City level systems will require huge computing resources with great variance at peak and off-peak times. Therefore, cloud computing will be best to deploy smart city services. There are numerous studies that propose different models to implement smart city services on cloud computing. For example, Alazawi et al. [47] have proposed an intelligent disaster management system based on cloud enabled vehicular networks. Suciu et al. [48] gave framework to automatically manage, analyze, and control data of varied characteristics in smart city context by distributed cloud-based services.

Cloud computing has been extended with fog computing to address its latency and other challenges. Fog computing "is an architecture that uses edge devices to carry out a substantial amount of computation, storage, communication locally and routed over the internet backbone" [49]. Other concepts related to fog computing include cloudlets and edge computing. Muhammed et al. [25] address the network latency, bandwidth, and reliability challenges of mobile cloud computing

in delivering services for anytime, anywhere capture and analyses of patients' data required by personalized healthcare systems. They propose UbeHealth, a ubiquitous healthcare framework that leverages edge computing, deep learning, big data, high-performance computing (HPC), and the Internet of Things (IoT) to address the aforementioned mobile cloud computing challenges. The framework comprises three main components and four layers, which enable provision of an enhanced network quality of service (QoS). The authors develop a proof-of-concept UbeHealth system based on the framework and evaluate its performance using three widely used datasets. A mobile computing system that provides enhanced mobility information through fog computing is proposed in [50]. A middleware fog computing platform is used along with a mobile application and a backend cloud-based big data analysis system to provide timely information to the systems users.

Twalbeh et al. [51] propose a mobile cloud computing model for healthcare applications that uses cloudlets to allow the mobile device users to connect directly to cloud resources using cheaper technologies such as WiFi. A user is connected to the enterprise cloud only if the service is not available at the cloudlet layer. Networked healthcare and the role of mobile cloud computing and big data analytics in its enablement are discussed in [52]. The authors introduce and analyze a cloudlet-based mobile cloud computing infrastructure to be used for healthcare big data applications.

1.4.2 Internet of Things

One of the most important aspects of smart city is the digital sensing and actuation of real-world entities. Whole city is provided with the capability to be integrated with the ICT infrastructure. To facilitate this, entities or objects need to be uniquely identifiable, wirelessly connected, able to send and receive instruction and data, etc. Internet of things (IoT) is a new paradigm of internet that addresses these challenges and extends internet to principally non-computing entities. In the most simpler terms, it may be defined as "The worldwide network of interconnected objects uniquely addressable based on standard communication protocols" [53].

An elaborative working definition of IoT is given by IEEE "Internet of Things envisions a self-configuring, adaptive, complex network that interconnects 'things' to the Internet through the use of standard communication protocols. The interconnected things have physical or virtual representation in the digital world, sensing/actuation capability, a programmability feature and are uniquely identifiable. The representation contains information including the thing's identity, status, location or any other business, social or privately relevant information. The things offer services, with or without human intervention, through the exploitation of unique identification, data capture and communication, and actuation capability. The service is exploited through the use of intelligent interfaces and is made available anywhere, anytime, and for anything taking security into consideration" [54].

Zanella et al. [55] gave the technologies, architectures, and protocols for the urban IoT. They also explained the case of Padova Smart City project as a proof-of-concept deployment. The smart city IoT network will be operationalized through web services. The services could be several in domains such as traffic, waste, electricity, parking, and health. Irfan and Ahmad [56] have reviewed the existing IoT application in the healthcare area. They summarized the prevalent architectural model and technologies into three layers such as things layer, intermediate (middleware or gateway) layer, and backend computing (cloud computing) layer. Things layer consists of real-world entities laced with sensors and actuators. Intermediate layer uses multi-agent, SAO, RESTful, or fog computing technologies to offer quick pre-processing and communication to the physical objects. Backend layer working on cloud computing paradigm offers high performance computing and big data support.

Alam et al. [57] study the use of eight well-known data mining algorithms for IoT including deep learning. Their study uses three widely used real datasets and shows C4.5 and C5.0 to have better accuracy, memory efficiency, and higher processing speeds. Deep learning is found to have better accuracy but requires relatively intensive computations. A survey of data fusion techniques for IoT is provided in [58]. They focus particularly on mathematical methods for data fusion including artificial intelligence, probabilistic, and theory of belief methods. They also review specific IoT environments including nonlinear, object tracking and heterogeneous environments. The challenges, opportunities, and areas of future developments for data fusion techniques for IoT are also discussed. Muhammed et al. [59] propose a new fault-tolerant routing technique for hierarchical sensor networks in IoT environments. A taxonomy for fault-tolerant techniques in IoT environments is given along with a quantitative comparison of some existing fault-tolerant techniques. A distance learning system is proposed in [60] which analyzes user activity real data acquired from IoT devices to understand and predict their spatio-temporal activities and other patterns of the user. This analysis is used further in optimizing the mode (text, voice or video) of module delivery to the users.

1.4.3 Big Data, High-Performance Computing (HPC), and their Convergence

Big data refers to the *"emerging technologies that are designed to extract value from data having four Vs characteristics; volume, variety, velocity and veracity"* [61]. Big data is being generated from various sources such as Internet of Things (IoT) and social media, and is being used in many application areas, see e.g., [62–64]. Arfat et al. [63] draw attention to the need for developing new technologies for big data processing because big data cannot be processed by traditional tools and technologies due to their volume, velocity, veracity, and variety properties. They focus on graph computing that is becoming increasingly popular to model real-world problems, which are typically large and, hence, give rise to large graphs. These large graphs could be analyzed and solved using big data technologies. They

explore the performance of single source shortest path graph computations using the Apache Spark big data platform. They use the United States road network data, modeled as graphs, and calculate shortest paths between vertices. The experiments are performed on the Aziz supercomputer, a Top500 machine. They solve problems of varying graph sizes, i.e., various states of the USA, and analyze Spark's parallelization behaviour. They report that the speedup is expectedly dependent on both the size of the data and the number of parallel nodes.

A study on the use of big data for gaining insight into the experiences of citizens with government services is undertaken in [65]. The authors use Twitter data to evaluate the various services provided by the Ministry of Education in Saudi Arabia. Specifically, they investigate the users' opinions and expectations regarding the new university system launched by the ministry. Their aim with this study is to help the decision makers in the governments to identify the gaps between the government's expectations and the needs of the users informed by their study reflected in users' opinions and expectations.

The transformative potential of big data on urban transportations is investigated in [64]. The authors propose a theoretical framework that brings together big data, autonomous vehicles, and car free urban environment to improve inefficiencies of transportations and logistics operations through optimizations of shared resource capacity. The proposed framework is refined in a Markovian model to investigate several big data scenarios matching the demands of transportation operations of freight and people mobility with shared capacity of urban resources. Rapid, constant and extreme urbanization have created acute stress on urban infrastructure and quality of life. The authors claim their work to be "an initial first step in building theory, knowledge and critical understanding of the social implications being posed by the growth in cities and the role that big data and smart cities could play in developing a resilient and sustainable city transportation system." The authors recommend the reorganization and optimization of transportation operations in order to lower the costs and carbon footprint by moving from "individual firms optimizing their own transportation supply to a shared collaborative load and resourced system." This work is an extension of their earlier work presented in [62] that focuses on the use of big data in healthcare and transport capacity sharing.

Suma et al. [66] propose a big data architecture and system to detect spatio-temporal events around the London city with the aim to improve urban logistics and planning. Specifically, they use big data and AI platforms including Hadoop, Spark, and Tableau, and the Google Maps Geocoding API to study Twitter data about London and locate events. They find and locate congestion around the London city. They also discover that, during a certain period, top third tweeted words were about job and hiring, leading them to locate the source of the tweets which happened to be originating from around the Canary Wharf area, UK's major financial center. They extend their earlier work in [67] and present an enhanced methodology, big data pipeline, and software tool based on machine learning. They empirically demonstrate that events can be detected automatically by analyzing data and detect the occurrence of multiple events, their locations and times, without any prior knowledge of the events.

Arfat et al. [68] propose a mobile computing system that enables smarter cities with enhanced mobility information through big data, fog and cloud computing technologies. The system includes a mobile application, a backend cloud-based big data analysis system, and a middleware platform based on fog computing. The system brings multiple cutting-edge technologies together to provide uniquely focused information on user mobility. The system proposes to pull in and provide information to the users about their travel locally, nationally, and internationally. More importantly, relevant information is pulled in from multiple news media and other sources. The UTiLearn system [24] is another example of big data system. It uses big data and other emerging technologies to provide enhanced development, management, and delivery of teaching and learning in smart society settings. We have discussed this in the smart city systems section.

The need to optimize supply chain activities in healthcare using big data is highlighted in [69]. The authors note that since medical equipment and devices generate massive amounts of data, big data analytics, which is proven to be helpful in forecasting and decision-making, can be a powerful tool to improve healthcare supply chains. They present a survey on the use of big data in healthcare supply chains and discuss the challenges, opportunities and future directions for big data enabled healthcare supply chains.

The role of big data, high-performance computing (HPC), and deep learning for disaster management is proposed in [70]. The authors extend the earlier work of Alazawi et al. [71] and use deep learning to predict urban traffic in disaster situations. They use Graphics Processing Units (GPUs) to address the compute-intensive nature of deep learning. They use open road traffic data made available by the Department for Transport, UK. They claim their work to be the first in applying deep learning for disaster management.

Numerous scientific, engineering, and smart city applications require the solution of sparse linear equation systems and Sparse Matrix Vector (SpMV) computations. For example, Markov chains have been used to model many smart city systems, see e.g., [72]. These problems require HPC techniques. Alyahya et al. [73] study SpMV performance on Intel Many Integrated Core (MIC) Architecture which has seen limited works in the past. Intel MIC and other many core architectures will be among the range of heterogeneous devices available in smart cities computational infrastructures and, hence, studying the behavior of smart city applications on such devices is of high importance.

High-performance computing (HPC) has traditionally been associated with tightly coupled supercomputing platforms and later on with cluster computing. However, broadly speaking, any techniques that aim to enhance computational performance by using the capacity of the underlying computational resources (hardware, software, caches, etc.), particularly using parallel or concurrent computing paradigms, can fall under HPC. Usman et al. [74] discuss the convergence of big data and HPC, reviewing the driving forces, challenges, current and future trends of the convergence. They note that the rise of HPDA (High Performance Data Analytics) has resulted in the expansion of HPC market in many new territories including big data. Farber [75] reviews a number of recent initiatives and trends on the HPC and big data convergence.

1.4.4 Smart City Applications and Systems

Apart from the architectural complexities and technological components, it is also necessary to organize the different essential services of a smart city system into a modular fashion. As described in the previous chapter, there are six major dimensions, into which the city must show performance. Therefore, smart city systems should identify different services in these dimensions that need to be digitized to tread the path of smartness. These modules can be termed as Smart Economy Management (SEM), Smart People Management (SPM), Smart Governance Management (SGM), Smart Mobility Management (SMM), Smart Environment Management (SEM), and Smart Living Management (SLM). We review below some proposals on smart city systems and services.

Many works have been proposed on the design and evaluation of disaster management systems for smart cities. These include, for example, a system architecture using cloud computing [47], a system using vehicular ad hoc networks (VANETs) [76], a system with specific focus on city evacuation strategies [77, 78], and other improvements to the earlier proposed disaster management systems [71]. These works were based on mathematical modeling and simulation studies of disaster-related scenarios. These works were extended in [70] using a deep learning based study.

Alomari et al. [79] present a study on the use of Twitter data for detecting road traffic conditions in the Jeddah city. They discover the most congested roads in Jeddah and the traffic relationship with the tweeting behavior. This study is carried out using tweets in the Arabic language on the SAP Hana platform. Suma et al. [66] note the emerging use of social media for sensing the information about the people and their spatiotemporal experiences around the living spaces. They use Twitter data to detect spatio-temporal events around the London city with the aim to improve urban logistics and planning. They extend in [67] their earlier work and present an enhanced methodology and software tool based on machine learning techniques. They empirically demonstrate that various conceptual, virtual, or physical events can be detected automatically by analyzing data and detect the occurrence of multiple events including "Underbelly Festival," "The Luna Cinema" and "London Notting Hill Carnival 2017," their locations and times, without any prior knowledge of the events.

A study on the computing single source shortest path routes for the United States road network data is presented in [63]. The authors model the United States road network as graphs and calculate shortest paths between vertices. An object recognition method for autonomous driving for smart cities called Decision Tree and Decision Fusion based Recognition System (D2TFRS) is proposed in [80].

Mehmood et al. [24] accentuate the changing landscape of the education industry worldwide due to the emergence of the Massive Open Online Course (MOOC) models, changing behaviors of digital learners, and the continuing gloomy global economy. Driven by the lacking sophistication of distance eTeaching and eLearning (DTL) systems, due to the challenges related to "data analysis and management, learner-system interactivity, system cognition, resource planning, agility, and scalability," they propose UTiLearn, a personalized ubiquitous eTeaching & eLearning

framework. UTiLearn "leverages Internet of Things, big data, supercomputing, and deep learning to provide enhanced development, management, and delivery of teaching and learning in smart society settings." They develop a proof-of-concept UTiLearn system based on the proposed framework and provide a detailed design, implementation, and evaluation of the UTiLearn system, including its five components, using 11 widely used datasets.

Al-dhubhani et al. [81] review cutting-edge technologies and their potential use for border security. They review literature related to the data acquisition technologies, storage, processing, analysis, and decision-making technologies of border security. They propose a smarter border security system along with ideas for future research and development.

Alamoudi et al. provide a review of DNA profiling methods and tools in [82]. DNA typing or profiling, as defined by them, "is a widely used practice in various forensic laboratories, used, for example, in sexual assault cases when the source of DNA mixture can combine different individuals such as the victim, the criminal, and the victim's partner." Highlighting the motivations for DNA profiling, they state that faster interpretations of DNA mixture profiles will expectedly open up new frontiers for this area in smart society applications. Khanum et al. [83] propose a novel fuzzy logic based framework for managing the complex tasks of smart farming in smart society scenarios. The framework is called Semantically Enriched Computational Intelligence (SECI). The authors discuss attributes of SECI that make it a suitable computational technique to be applied to various dimensions of smart farming. They describe three possible applications of SECI in smart farming that they plan to implement in the future. The implementation of one of the described applications of smart farming is discussed in detail along with its preliminary results.

Al-Dhubhani et al. [84] provide a review of location privacy research related to smart cities. They review smart city architectures, frameworks, and platforms, and discuss the extent to which the preservation of location privacy has been addressed. They claim based on the provided literature review that the preservation of location privacy has not received sufficient attention in smart city applications. They discuss the issues that should be addressed to improve the preservation of location privacy for smart city applications and propose a location privacy preservation system for smart city applications.

Numerous other smart city technologies and systems have been proposed, for example, related to mobile computing [85], emerging applications [86], healthcare and life sciences [25, 51, 69, 82], information systems [41, 87], and IoT-based smart applications [59]. Moreover, smart mobility is a key dimension of smart city designs and operations [88]. Plentiful approaches have been proposed to address transportation challenges and develop smart transportation infrastructures, see e.g., autonomic transport systems [89–91], vehicular networks (VANETs) and systems [92–95], emergency management system [78], simulations [96, 97], urban logistics [62, 72], big data [62–64], location based services [98], and social media based approaches [67, 99]. Some of these works on transportation are multidisciplinary and these have already been discussed in this section or other sections. A recent book has covered a number of topics related to Smart Societies, Infrastructure, Technologies, and Applications [100].

The smart city planner should develop the whole system utilizing the latest ICT such as web services and SoA, cloud computing, IoT, big data and semantic web. This will provide the needed scalability, resilience, and portability to address the city level complexity and challenges.

1.5 Structural Equation Modeling

The smart city domain is highly interdisciplinary and requires modeling different variables of interest including latent variables. First-generation statistical techniques such as regression, factor analysis, or cluster analysis use empirical data to confirm or identify the theoretical hypothesis. These techniques have certain limitations or make unrealistic assumption such as multiple independent variables regress to one dependent variable, variables to be observable, and the error free measurements of variables [101].

Structural equation modeling (SEM) defeats the limitations of first-generation statistical techniques and allows for the modeling of multiple independent constructs and multiple dependent construct in a holistic, systematic, and unified way [102]. Every variable is considered either exogenous (independent) or endogenous (dependent). And the latter is being explained by the relationships postulated in the model [103]. SEM employs two techniques covariance or variance to analyze the model. Partial least square (PLS) regression uses a variance-based approach.

In order to develop a model profound understating of theory is mandatory. So that the model structure is in sync with the theoretical structures available in the theory. The theory consists of theoretical concepts and derived concepts both defined as latent variables and an empirical concept defined as an indicator variable. Moreover, these concepts are linked with non-observational hypothesis (linking theoretical concepts with theoretical concepts), theoretical definitions (linking theoretical concepts with derived concepts), and correspondence rules (linking theoretical or derived concepts with empirical concepts) [104]. This section will illustrate upon the basics of SEM more specifically PLS SEM and its application in the smart city logistics domain.

1.5.1 Reflective Versus Formative Constructs

Reflective constructs cause the items or measurements whereas formative constructs are caused by the items or measurements [105]. In the case of reflective constructs any change in the latent construct is reflected in the measurement and measurement error is associated with the measurements [105]. On the other hand, formative constructs are derived from measurements and the measurement error is introduced at the level of construct [105].

LISREL, EQS and AMOS, covariance-based SEM doesn't support formative constructs [102] whereas variance-based, PLS SEM supports both formative and reflective constructs [102]. Moreover, PLS doesn't make any assumption about the population or scale of measurement [106]. Therefore, it works with nominal and higher scale and with no assumptions of the distribution [101].

1.5.2 Partial Least Squares (PLS) Regression

PLS utilizes the concept of maximization of variance of the dependent variables explained by the independent variables in contrast to the reconstructing of the covariance matrix [101]. The PLS model consists of a structure showing all the latent or unobservable constructs along with their measurements, items, or indicators. Secondly it consists of paths showing the relationships among them. Finally, it has an additional component that defines the weight relation used to estimate the unobservable variables [101].

The order of computation goes as follows first the weight relations are calculated, next using these weights the case values are estimated that is weighted average of indicators, finally the structural relations are calculated with the help of a set of regression equations [106]. Therefore, the most important step is the estimation of weight relations since all successive steps depend on it. In a more simplistic approach, one may give equal weights to all the indicators, but it will be difficult to get theoretical support for the same [101]. Moreover, it will also undermine the more reliable indicators supposed to get higher weights [107].

PLS employs two-step outside and inside approximation, iteratively until the case values converge. For outside approximation, the latent variable is estimated with their respective indicators with the help of regression for formative constructs as proposed in this research [101]. And for inside approximation, the case value is weighted with the help of neighboring latent variables using centroid, factor or path weighing Scheme [101]. Moreover, the problem of consistency is removed if the sample size and the number of indicators approach infinity.

1.5.3 Construct Reliability and Validity

Outside approximation more known as an outer model or measurement model provides the reliability and validity of blocks of manifest variables [108]. There are five criteria for the assessment of outer model for the reflexive constructs, namely indicator reliability, internal consistency reliability, convergent validity, and discriminant validity at indicator and construct levels [108].

Indicator reliability or the reliability of the manifest variable is achieved when factor loading is more than 0.7 [109]. Composite reliability or Cronbach's alpha more than 0.6 is required for the internal consistency reliability or reliability of block of manifest variable [109]. Convergent validity, a measure indicating manifest variables represent the underlying construct, is assessed by average variance extracted (AVE) and should be more than 0.5 [109]. Discriminant validity at the construct level is a measure of the extent to which constructs don't correlate with other constructs. It is estimated with the Fornell–Larcker criterion that is construct's AVE should be higher than its squared correlation with other constructs [110]. Whereas, the indicator level discriminant validity is established when manifest variable loads highest on the mapped construct.

For the formative constructs, the indicator reliability is meaningless due to the assumption of error free measures [109]. Three criteria are used to assess the measurement model for the formative constructs, namely indicators relative contribution to the construct, significance of weight, and multicollinearity [108]. Of these last two are more important to establish since they decide as to which indicator will enter the model. Bootstrapping method is used to assess the significance of the estimated indicators weight [109]. The multicollinearity among formative constructs is estimated with the help of variance inflation factor (VIF) and a value more than 10 shows critical multicollinearity [109]. However, the value of VIF less than 3.3 is considered excellent [111]. Moreover, the value of condition number for the construct below 30 signifies the absence of any collinearity [112].

1.5.4 Assessment of Inner or Structural Model

Once the reliable and valid outer model is achieved, the inner model is estimated. The coefficient of determination R^2 value of endogenous variable is an essential criterion [109]. The values of R^2 0.67, 0.33, or 0.19 define the endogenous latent variable to be substantial, moderate, or weak, respectively [113]. The lesser value of R^2 is acceptable where one or two exogenous variables explain endogenous variable. The second criterion is the path coefficient assessed for value, sign, and significance. The significance is estimated with the help of bootstrapping method.

The third criterion is the effect size f^2. The values of f^2 0.02, 0.15, or 0.35 can be considered as a measure of weak, medium, or large effect at the structural level [109]. Last but not the least measure is the predictive capability of the structural model. This is estimated with the help of predictive relevance Q^2, that measures how well observed values are reconstructed by model and its parameters [108]. The value of Q^2 is calculated by blindfolding procedure. A value above zero indicates that the observed values are close to predicted values and proves the predictive relevance of the model [109].

1.5.5 PLS Applications in Smart City

In the previous studies [87, 114], authors have applied PLS SEM in the context of enterprise systems and future city logistics. A success predictive model that will ensure the benefits of enterprise systems was developed using PLS SEM. Further the study synthesized the benefits of ES in different dimensions of future city logistics. Abbasi et al. [115] have used hybrid support vector machine and PLS model to forecast municipal solid waste forecasting. Another study has used PLS regression to analyze the model of RFID adoption in public transportation services using innovation diffusion theory [116].

The discipline of smart city is highly interdisciplinary and will have multiple phenomenon or constructs, observable and latent. As pointed out earlier that latent variables can be reflexive or formative, PLS SEM will be most suited to study these variables in order for theory building in smart city domain. For instance, authors believe that connectedness leads to integration, integration leads to dynamism, dynamism leads to smartness and finally smartness, will further give impetus for more connectedness. This proposed model herein of smart city systems adoption shown in Fig. 1.2 may also be established using PLS SEM or any other system of variables.

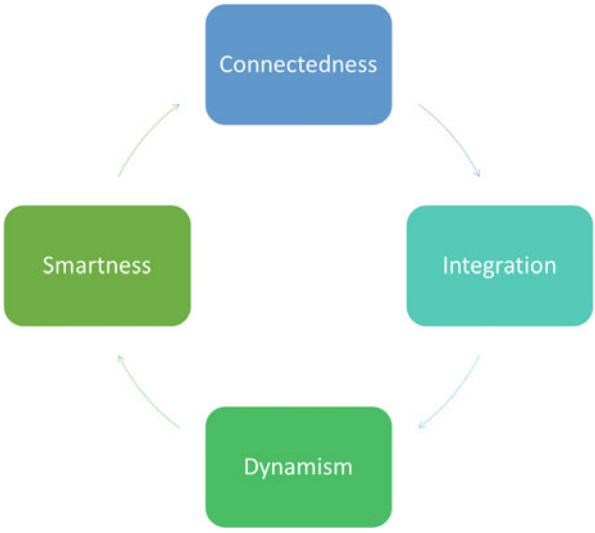


Fig. 1.2 Smart city systems adoption model

1.6 Conclusion

The smart city microcosm should be defined by all the stakeholders of the society unlike corporate organizations. People from every strata and trade should collectively define, design, develop, and deliver the smart city. That's the only way to avoid gentrification. No city in the world would dare not to aspire to become smart city and if the purpose is to beat the competition then cities should have to focus on to invest heavily on differential or strategic assets such as knowledge infrastructure along with other necessary commodity infrastructures.

This chapter has done extensive literature review on the concept of smart city in order to derive the systems perspective of smart city. Nonetheless, the alignment of ICT with the true concept of smart city, strategically and operationally, needs to be exemplified in the definition of smart city systems. Authors have defined the smart city systems as "Smart City System is a convergence of integrate systems supporting city processes reengineered on sustainability principles and utilizing state of the art technology to advance frontiers of knowledge, economy and life in a society." The field of enterprise systems has also been explained as SCS will make use of different aspects of this technology. Similarly, the emerging areas of ICT such as service based distributed computing, IoT, Big Data and High-Performance Computing, have been illustrated upon. Also, there are numerous applications in the smart city context such as disaster management, traffic management, spatio-temporal experiences of citizens, and DNA profiling for crime prevention. SEM section gives important insight into concepts and applications of PLS SEM essential for theory building in this domain.

ICT must be utilized in every aspect of smart city as it can bring true connectedness that leads to integration and integration leads to dynamism and dynamism leads to smartness (Fig. 1.2). Smartness will further give impetus for more connectedness and hence the cycle of innovation continues to realize best in class smart city.

References

1. Komninos, N.: Intelligent cities: variable geometries of spatial intelligence. Intell. Build. Int. **3**, 172–188 (2011)
2. Hall, R.E., Bowerman, B., Braverman, J., Taylor, J., Todosow, H.: The vision of a smart city. 2nd Int. Life Ext. Technol. Work. **7**, (2000)
3. Giffinger, R.: Smart cities ranking of European medium-sized cities. October. **16**, 13–18 (2007)
4. Hollands, R.G.: Will the real smart city please stand up? City. **12**, 303–320 (2008)
5. Washburn, D., Sindhu, U., Balaouras, S., Dines, R.A., Hayes, N., Nelson, L.E.: Helping CIOs understand "Smart City" initiatives. Growth. **17**, 1–17 (2009)
6. Harrison, C., Eckman, B., Hamilton, R., Hartswick, P., Kalagnanam, J., Paraszczak, J., Williams, P.: Foundations for smarter cities. IBM J. Res. Dev. **54**, 1–16 (2010)

7. Chen, T.: Smart grids, smart cities need better networks. Journals & Magazines. **24**(1), 2–3 (2010)
8. Caragliu, A., Del Bo, C., Nijkamp, P.: Smart cities in Europe. J. Urban Technol. **18**, 65–82 (2011)
9. Velosa, A., Tratz-Ryan, B., Anavitarte, L., Fernando, H.: Market Trends: Smart Cities Are the New Revenue Frontier for Technology Providers. (2011). https://www.gartner.com/doc/1615214/market-trends-smart-cities-new
10. Nam, T., Pardo, T. a.: Conceptualizing smart city with dimensions of technology, people, and institutions. In: Proceedings of the 12th Annual International Digital Government Research Conference on Digital Government Innovation in Challenging Times - dg.o '11. p. 282 (2011)
11. Thite, M.: Smart cities: implications of urban planning for human resource development. Hum. Resour. Dev. Int. **14**, 623–631 (2011)
12. Rios, P.: Creating" The Smart City". 1–126 (2012)
13. Thuzar, M.: Urbanization in Southeast Asia: developing smart cities for the future? Reg. Econ. Outlook. **96**, 100 (2012)
14. Barrionuevo, J.M., Berrone, P., Ricart, J.E.: Smart cities, Sustainable Progress. IESE Insight. **14**, 50–57 (2012)
15. Telefónica, F., CRETU, G.L.: Smart cities design using event-driven paradigm and semantic web. Inform. Econ. **16**, 57–67 (2012)
16. Kourtit, K., Nijkamp, P.: Smart cities in the innovation age. Innov. Eur. J. Soc. Sci. Res. **25**, 93–95 (2012)
17. Kourtit, K., Nijkamp, P., Arribas, D.: Smart cities in perspective – a comparative European study by means of self-organizing maps. Innov. Eur. J. Soc. Sci. Res. **25**, 229–246 (2012)
18. Lazaroiu, G.C., Roscia, M.: Definition methodology for the smart cities model. Energy. **47**, 326–332 (2012)
19. Lombardi, P., Giordano, S., Farouh, H., Yousef, W.: Modelling the smart city performance. Innov. Eur. J. Soc. Sci. Res. **25**, 137–149 (2012)
20. Bakici, T., Almirall, E., Wareham, J.: A Smart City initiative: the case of Barcelona. J. Knowl. Econ. **4**, 135–148 (2013)
21. Zygiaris, S.: Smart City reference model: assisting planners to conceptualize the building of Smart City innovation ecosystems. J. Knowl. Econ. **4**, 217–231 (2013)
22. Marsal-Llacuna, M.: Lessons in urban monitoring taken from sustainable and livable cities to better address the Smart Cities initiative. Forecast. Soc. (2015)
23. Albino, V., Berardi, U., Dangelico, R.M.: Smart cities: definitions, dimensions, performance, and initiatives. J. Urban Technol. **22**, 3–21 (2015)
24. Mehmood, R., Alam, F., Albogami, N.N., Katib, I., Albeshri, A., Altowaijri, S.: UTiLearn: a personalised ubiquitous teaching and learning system for smart societies. IEEE Access. **3536**, 1–22 (2017)
25. Muhammed, T., Mehmood, R., Albeshri, A., Katib, I.: UbeHealth: a personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities. IEEE Access. **6**, 32258–32285 (2018)
26. Gomolski, B., Grigg, J., Potter, K.: IT spending and staffing survey results. Stamford, CT (2001)
27. Carr, N.G.: IT doesn't matter. Educ. Rev. **38**, 24–38 (2003)
28. Terry Anthony Byrd, D.E.T.: Measuring the flexibility of information technology infrastructure: Exploratory analysis of a construct. J. Manag. Inf. Syst. 17, 167–208 (2000)
29. Chanopas, A., Krairit, D., Ba Khang, D.: Managing information technology infrastructure: a new flexibility framework. Manag. Res. News. **29**, 632–651 (2006)
30. Weill, P., Subramani, M., Broadbent, M.: Building IT infrastructure for strategic agility. MIT Sloan Manag. Rev. **44**, 57–65 (2002)
31. Davenport, T.H.: Putting the enterprise into the enterprise system. Harv. Bus. Rev. **76**, 121–131 (1998)
32. Xue, Y., Liang, H., Boulton, W.R., Snyder, C.A.: ERP implementation failures in China: case studies with implications for ERP vendors. Int. J. Prod. Econ. **97**, 279–295 (2005)

33. Hwa Chung, S., Snyder, C.A.: ERP adoption: a technological evolution approach. Int. J. Agil. Manag. Syst. **2**, 24–32 (2000)
34. Davenport, T.H., Brooks, J.D.: Enterprise systems and the supply chain. J. Enterp. Inf. Manag. **17**, 8–19 (2004)
35. Møller, C.: ERP II: a conceptual framework for next-generation enterprise systems? J. Enterp. Inf. Manag. **18**, 483–497 (2005)
36. Klaus, H., Rosemann, M., Gable, G.G.: What is ERP? Inf. Syst. Front. **2**, 141–162 (2000)
37. Sathish, S., Pan, S., Raman, K.: A stakeholder perspective of enterprise systems. In: PACIS 2003 Proceedings. pp. 669–682. , Adelaide, South Australia (2003)
38. Davenport, T.H.: Process Innovation: Reengineering Work through Information Technology. Harvard Business Press, Boston, MA (1993)
39. van Slooten, K., Yap, L.: Implementing ERP information systems using SAP. In: AMCIS 1999 proceedings. p. 81. Milwaukee (1999)
40. Motiwalla, L.F., Thompson, J.: Enterprise systems for management. Pearson, Boston, MA (2012)
41. Ahmad, N., Mehmood, R.: Enterprise systems: are we ready for future sustainable cities. Supply Chain Manag. **20**, 264–283 (2015)
42. Davenport, T.H., Short, J.E., others: The new industrial engineering: information technology and business process redesign. Sloan Manage. Rev. 11–27 (1990)
43. Shen, H., Wall, B., Zaremba, M., Chen, Y., Browne, J.: Integration of business modelling methods for enterprise information system analysis and user requirements gathering. Comput. Ind. **54**, 307–323 (2004)
44. Curran, T., Keller, G., Ladd, A.: SAP R/3 Business Blueprint: Understanding the Business Process Reference Model. Prentice-Hall, Inc., Englewood Cliffs, NJ (1997)
45. Mell, P., Grance, T.: The NIST Definition of Cloud Computing, http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf
46. Ahmad, N.: Cloud computing: Technology, security issues and solutions. In: Anti-Cyber Crimes (ICACC), 2017 2nd International Conference on. pp. 30–35 (2017)
47. Alazawi, Z., Altowaijri, S., Mehmood, R., Abdljabar, M.B.: Intelligent disaster management system based on cloud-enabled vehicular networks. In: ITS Telecommunications (ITST), 2011 11th International Conference on. pp. 361–368 (2011)
48. Suciu, G., Vulpe, A., Halunga, S., Fratu, O., Todoran, G., Suciu, V.: Smart cities built on resilient cloud computing and secure internet of things. In: Control Systems and Computer Science (CSCS), 2013 19th International Conference on. pp. 513–518 (2013)
49. WikiPedia: Fog Computing
50. Arfat, Y., Aqib, M., Mehmood, R., Albeshri, A., Katib, I., Albogami, N., Alzahrani, A.: Enabling Smarter Societies through Mobile Big Data Fogs and Clouds. In: Procedia Computer Science (2017), 109, 1128
51. Tawalbeh, L.A., Bakhader, W., Mehmood, R., Song, H.: Cloudlet-based mobile cloud computing for healthcare applications. In: 2016 IEEE Global Communications Conference, GLOBECOM 2016 - Proceedings (2016)
52. Tawalbeh, L.A., Mehmood, R., Benkhlifa, E., Song, H.: Mobile cloud computing model and big data analysis for healthcare applications. IEEE Access. **4**, 6171–6180 (2016)
53. Bassi, A., Horn, G.: Internet of things in 2020: a roadmap for the future. Eur. Comm. Inf. Soc. Media. **22**, 97–114 (2008)
54. Minerva, R., Biru, A., Rotondi, D.: Towards a definition of the internet of things (IoT). IEEE Internet Initiat. **1**, 1–86 (2015)
55. Zanella, A., Bui, N., Castellani, A., Vangelista, L., Zorzi, M.: Internet of things for smart cities. IEEE Internet Things J. **1**, 22–32 (2014)
56. Irfan, M., Ahmad, N.: Internet of Medical Things: Architectural Model, Motivational Factors and Impediments. In: 2018 15th Learning and Technology Conference (L&T). pp. 6–13 (2018)
57. Alam, F., Mehmood, R., Katib, I., Albeshri, A.: Analysis of Eight Data Mining Algorithms for Smarter Internet of Things (IoT). In: Procedia Computer Science. pp. 437–442 (2016)

58. Alam, F., Mehmood, R., Katib, I., Albogami, N.N., Albeshri, A.: Data fusion and IoT for smart ubiquitous environments: a survey. IEEE Access. **5**, 9533–9554 (2017)
59. Muhammed, T., Mehmood, R., Albeshri, A.: Enabling reliable and resilient IoT based smart city applications. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, vol 224. pp. 169–184. Springer, Cham (2018)
60. Mehmood, R., Alam, F., Albogami, N.N., Katib, I., Albeshri, A., Altowaijri, S.M.: UTiLearn: a personalised ubiquitous teaching and learning system for smart societies. IEEE Access. **5**, 2615–2635 (2017)
61. Mehmood, R., Faisal, M.A., Altowaijri, S.: Future networked healthcare systems: a review and case study. In: Boucadair, M., Jacquenet, C. (eds.) Handbook of Research on Redesigning the Future of Internet Architectures, pp. 531–558. IGI Global, Hershey, PA (2015)
62. Mehmood, R., Graham, G.: Big data logistics: a health-care transport capacity sharing model. Procedia Comput. Sci. **64**, 1107–1114 (2015)
63. Arfat, Y., Mehmood, R., Albeshri, A.: Parallel shortest path graph computations of United States road network data on apache spark. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 323–336. Springer, Cham (2018)
64. Mehmood, R., Meriton, R., Graham, G., Hennelly, P., Kumar, M.: Exploring the influence of big data on city transport operations: a Markovian approach. Int. J. Oper. Prod. Manag. **37**, 75–104 (2017)
65. Alsulami, Mashael; Mehmood, R.: Sentiment Analysis Model for Arabic Tweets to Detect Users' Opinions about Government Services in Saudi Arabia: Ministry of Education as a case study. In: Al Yamamah Information and Communication Technology Forum. pp. 1–8. , Riyadh (2018)
66. Suma, S., Mehmood, R., Albugami, N., Katib, I., Albeshri, A.: Enabling Next Generation Logistics and Planning for Smarter Societies. Procedia - Procedia Comput. Sci. 1–6 (2017)
67. Suma, S., Mehmood, R., Albeshri, A.: Automatic Event Detection in Smart Cities Using Big Data Analytics. In: International Conference on Smart Cities, Infrastructure, Technologies and Applications SCITA 2017: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 111–122. Springer, Cham (2018)
68. Arfat, Y., Aqib, M., Mehmood, R., Albeshri, A., Katib, I., Albogami, N., Alzahrani, A.: Enabling smarter societies through Mobile big data fogs and clouds. Procedia Comput. Sci. **109**, 1128–1133 (2017)
69. Alotaibi, S., Mehmood, R.: Big data enabled healthcare supply chain management: Opportunities and challenges. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering (LNICST), Volume 224. pp. 207–215. Springer, Cham (2018)
70. Aqib, M., Mehmood, R., Albeshri, A., Alzahrani, A.: Disaster management in smart cities by forecasting traffic plan using deep learning and GPUs. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 139–154 (2018)
71. Alazawi, Z., Alani, O., Abdljabar, M.B., Altowaijri, S., Mehmood, R.: A smart disaster management system for future cities. In: Proceedings of the 2014 ACM international workshop on Wireless and mobile technologies for smart cities - WiMobCity '14. pp. 1–10. ACM Press, New York, New York, USA (2014)
72. Mehmood, R., Lu, J.A.: Computational Markovian analysis of large systems. J. Manuf. Technol. Manag. **22**, 804–817 (2011)
73. Alyahya, H., Mehmood, R., Katib, I.: Parallel sparse matrix vector multiplication on intel MIC: Performance analysis. In: Smart Societies, Infrastructure, Technologies and Applications, SCITA 2017, Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 306–322. Springer, Cham (2018)

74. Usman, S., Mehmood, R., Katib, I.: Big data and HPC convergence: The cutting edge and outlook. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 11–26. Springer, Cham (2018)
75. Rob Farber: The Convergence of Big Data and Extreme-Scale HPC
76. Alazawi, Z., Abdljabar, M.B., Altowaijri, S., Vegni, A.M., Mehmood, R.: ICDMS: An intelligent cloud based disaster management system for vehicular networks. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), LNCS, Volume 7266. pp. 40–56. Springer, Vilnius, Lithuania (2012)
77. Alazawi, Z., Alani, O., Abdljabar, M.B., Mehmood, R.: Transportation evacuation strategies based on VANET disaster management system. Procedia Econ. Financ. **18**, 352–360 (2014)
78. Alazawi, Z., Alani, O., Abdljabar, M.B., Mehmood, R.: An intelligent disaster management system based evacuation strategies. In: 2014 9th International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2014. pp. 673–678 (2014)
79. Alomari, E., Mehmood, R.: Analysis of tweets in Arabic language for detection of road traffic conditions. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST. pp. 98–110. Springer, Cham (2018)
80. Alam, F., Mehmood, R., Katib, I.: D2TFRS: An object recognition method for autonomous vehicles based on RGB and spatial values of pixels. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 155–168. Springer, Cham (2018)
81. Al-dhubhani, Raed; Al Shehri, Waleed; Mehmood, Rashid; Katib, Iyad; Algarni, Abdullah; Altowaijri, S.: Smarter Border Security: A Technology Perspective. In: 1st International Symposium on Land and Maritime Border Security and Safety, Saudi Arabia. pp. 131–143. , Jeddah (2017)
82. Alamoudi, E., Mehmood, R., Albeshri, A., Gojobori, T.: DNA profiling methods and tools: A review. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 216–231 (2018)
83. Khanum, A., Alvi, A., Mehmood, R.: Towards a semantically enriched computational intelligence (SECI) framework for smart farming. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 247–257. Springer, Cham (2018)
84. Al-Dhubhani, R., Mehmood, R., Katib, I., Algarni, A.: Location privacy in smart cities era. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST Volume 224. pp. 123–138. Springer, Cham (2018)
85. Tawalbeh, L., Basalamah, A., Mehmood, R., Tawalbeh, H.: Greener and Smarter Phones for Future Cities: Characterizing the Impact of GPS Signal Strength on Power Consumption. IEEE Access. PP, 1–1 (2016)
86. Graham, G., Mehmood, R.: The strategic prototype "crime-sourcing" and the science/science fiction behind it. Technol. Forecast. Soc. Change. **84**, 86–92 (2014)
87. Ahmad, N., Mehmood, R.: Enterprise systems and performance of future city logistics. Prod. Plan. Control. **27**, 500–513 (2016)
88. Büscher, M., Coulton, P., Efstratiou, C., Gellersen, H., Hemment, D., Mehmood, R., Sangiorgi, D.: Intelligent mobility systems: Some socio-technical challenges and opportunities. In: Communications Infrastructure. Systems and Applications in Europe, Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST 16. pp. 140–152 (2009)
89. Schlingensiepen, J., Nemtanu, F., Mehmood, R., McCluskey, L.: Autonomic Transport Management Systems—Enabler for Smart Cities, Personalized Medicine, Participation and Industry Grid/Industry 4.0. In: Intelligent Transportation Systems – Problems and Perspectives, Volume 32 of the series Studies in Systems, Decision and Control. pp. 3–35. Springer International Publishing (2016)
90. Schlingensiepen, J., Mehmood, R., Nemtanu, F.C.: Framework for an autonomic transport system in smart cities. Cybern. Inf. Technol. **15**, 50–62 (2015)

91. Schlingensiepen, J., Mehmood, R., Nemtanu, F.C., Niculescu, M.: Increasing sustainability of road transport in European cities and metropolitan areas by facilitating autonomic road transport systems (ARTS). In: Wellnitz, J., Subic, A., Trufin, R. (eds.) Sustainable Automotive Technologies 2013 Proceedings of the 5th International Conference ICSAT 2013, pp. 201–210. Springer International Publishing, Ingolstadt, Germany (2014)
92. Mehmood, R., Nekovee, M.: Vehicular AD HOC and grid networks: Discussion, design and evaluation. In: 14th World Congress on Intelligent Transport Systems, ITS 2007. pp. 1555–1562 (2007)
93. Gillani, S., Shahzad, F., Qayyum, A., Mehmood, R.: A Survey on Security in Vehicular Ad Hoc Networks. Springer, Berlin (2013)
94. Alvi, A., Greaves, D., Mehmood, R.: Intra-vehicular verification and control: A two-pronged approach. In: 7th IEEE International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2010. pp. 401–405 (2010)
95. Nabi, Z., Alvi, A., Mehmood, R.: Towards standardization of in-car sensors. In: lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics). LNCS Volume. **6596**, 216–223 (2011)
96. Ayres, G., Mehmood, R.: On discovering road traffic information using virtual reality simulations. In: 11th International Conference on Computer Modelling and Simulation, UKSim 2009. pp. 411–416 (2009)
97. Mehmood, R.: Towards understanding intercity traffic interdependencies. In: 14th World Congress on Intelligent Transport Systems, ITS 2007. pp. 1793–1799 (2007)
98. Ayres, G., Mehmood, R.: LocPriS: A security and privacy preserving location based services development framework. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 566–575. Springer (2010)
99. Suma, S., Mehmood, R., Albugami, N., Katib, I., Albeshri, A.: Enabling next generation logistics and planning for smarter societies. Procedia Comput. Sci. **109**, 1122–1127 (2017)
100. Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.): Smart Societies, Infrastructure, Technologies and Applications, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST), Volume 224. Springer International Publishing, Cham (2018)
101. Haenlein, M., Kaplan, A.M.: A beginner's guide to partial least squares analysis. Underst. Stat. **3**, 283–297 (2004)
102. Gefen, D., Straub, D., Boudreau, M.-C.: Structural equation modeling and regression: guidelines for research practice. Commun. Assoc. Inf. Syst. **4**, 7 (2000)
103. Diamantopoulos, A.: Modelling with LISREL: a guide for the uninitiated. J. Mark. Manag. **10**, 105–136 (1994)
104. Bagozzi, R.P., Phillips, L.W.: Representing and testing organizational theories: a holistic construal. Adm. Sci. Q. **27**, 459–489 (1982)
105. Freeze, R.D., Raschke, R.L.: An Assessment of Formative and Reflective Constructs in IS Research. In: ECIS. pp. 1481–1492 (2007)
106. Fornell, C., Bookstein, F.L.: Two structural equation models: LISREL and PLS applied to consumer exit-voice theory. J. Mark. Res. **19**, 440–452 (1982)
107. Chin, W.W., Marcolin, B.L., Newsted, P.R.: A partial least squares latent variable modeling approach for measuring interaction effects: results from a Monte Carlo simulation study and an electronic-mail emotion/adoption study. Inf. Syst. Res. **14**, 189–217 (2003)
108. Ismail, I.R., Hamid, R.A., Idris, F.: PLS application in Journals of Operations Management: a review. In: Proceedings of Global Conference on Operations and Supply Chain Management. pp. 1–6 (2012)
109. Henseler, J., Ringle, C.M., Sinkovics, R.R.: The use of partial least squares path modeling in international marketing. In: New challenges to international marketing. pp. 277–319. Emerald Group Publishing Limited (2009)
110. Jnr, H.J.F., Money, A.H., Samouel, P., Page, M.: Research Methods for Business, UK Edition. Wiley, West Sussex, England (2007)

111. Diamantopoulos, A., Siguaw, J.A.: Formative versus reflective indicators in organizational measure development: a comparison and empirical illustration. Br. J. Manag. **17**, 263–282 (2006)
112. Hair, J.F., Anderson, R.E., Tatham, R.L., Black, W.C.: Multivariate Date Analysis with Readings, (Engleweood Cliffs, NJ). Prentice Hall, Englewood Cliffs, NJ (1995)
113. Chin, W.W.: The partial least squares approach to structural equation modeling. Mod. Methods Bus. Res. **295**, 295–336 (1998)
114. Ahmad, N.: Adoption, Implementation and Usage of Enterprise System: An Empirical Study, (2012)
115. Abbasi, M., Abduli, M.A., Omidvar, B., Baghvand, A.: Forecasting municipal solid waste generation by hybrid support vector machine and partial least square model. Int. J. Environ. Res. **7**, 27–38 (2013)
116. Liu, Y., Yang, Y., Wei, J., Wang, X.: An examination on RFID innovation diffusions in Chinese public intelligent transportation services. Int. J. Mob. Commun. **13**, 549–566 (2015)

Part I
Smart Transportation

Chapter 2
Sentiment Analysis of Arabic Tweets for Road Traffic Congestion and Event Detection

Ebtesam Alomari, Rashid Mehmood, and Iyad Katib

2.1 Introduction

More recently, Twitter has become a popular social platform to share traffic information. Mainly, Twitter can provide information about future events, the causes behind certain behavior, anomalies, and accidents, as well as the public feelings on a matter. Furthermore, there are specific, and official Twitter accounts created to report on traffic conditions and events in particular cities. These accounts generate useful sources of information for the followers. Hence, there is an enormous amount of traffic updates and information available in different Twitter accounts and can be freely obtained via the easy-to-access APIs [1].

Several researches have been proposed to monitor road traffic in different countries by analyzing text from different languages such as English and Chinese. However, the difficulty of performing such analysis in Arabic social media lies in the fact that the dialectical Arabic is used more than the formal Modern Standard Arabic (MSA), which produce new challenges for Arabic text classifications and Sentiment Analysis (SA) [2]. To the best of our knowledge, none of the existing works about sentiment analysis on Saudi dialect tweets have focused on traffic condition. Moreover, the existing analysis approaches for Arabic event detection did not focus on road traffic in Saudi Arabia. Further, they did not apply big data

E. Alomari (✉) · I. Katib
Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: EAlomari0011@stu.kau.edu.sa; iakatib@kau.edu.sa

R. Mehmood
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: RMehmood@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_2

technologies to properly handle such huge amounts of social data which required high processing speed, large storage, and other challenges.

Currently, road traffic congestion is one of the biggest problems in Saudi Arabia especially in large cities like Jeddah. Jeddah city is the second largest city in Saudi Arabia and arguably the most congested one. Further, Makkah is the Islam's holiest city, where millions of Muslims visit. The increasing number of vehicles and an enormous number of pilgrim visitors all year round have increased accidents and traffic jams in many major roads in this city. Moreover, the Kingdom accounts for over 40% of all active Twitter users in the Arab region [3]. By 2016, the number of Twitter users in Saudi Arabia had reached 4.99 million [4]. Hence, Twitter presents an excellent opportunity for extracting useful information. This raises the need for proposing a novel approach to analyze Arabic dialectical social data to monitor road traffic in Saudi Arabia.

In this paper, we extend our previous work [5] about analyzing and extracting traffic congestion information from Arabic tweets. In our previous work, we analyzed only negative tweets that refer to traffic jam and congestion where we designed the search queries to fetch tweets that contain specific negative traffic-related keywords. Subsequently, we extracted the traffic causes and the top congested roads and streets in Jeddah city.

In this work, we collect all traffic-related tweets regardless of the type (negative or positive). We fetch tweets about traffic in Jeddah and Makkah during Ramadan, which is the Islam's month of fasting. We chose this period to study the impact of this month on road traffic because in this month the traffic behavior and the road traffic rush hours change significantly. The main objectives and contributions of this paper can be summarized as follows:

- Improve our previous methodology by applying normalization on the extracted Arabic tokens.
- Provide a mechanism to detect events that could affect the traffic condition.
- Propose an approach for sentiment analysis to classify a driver's feeling and emotions.

Sentiment classification is one of the areas in which "big data" requires processing. Thus, we have built our approach on SAP HANA, which is an in-memory processing platform that can help to improve both the performance and the quality of the results. We analyzed the data by applying a lexicon-based approach. We have built lexicons (dictionaries) for Arabic and Saudi dialect words. The dictionaries include the most common words regarding traffic condition. The main goal is to classify traffic-related tweets into one of four sentiment classes (Strong positive, Positive, Strong negative, and Negative).

The rest of the paper is organized as follows. Section 2.2 reviews the related works. Section 2.3 illustrates the methodology. Section 2.4 discusses the results. Finally, we draw our conclusions in Sect. 2.5.

2.2 Literature Review

2.2.1 Transportation and Smart Cities

Traditional approaches for traffic measurement have relied on sensors that are buried under the road (such as inductive loops) or installed on roadside [6]. Additionally, many traffic monitoring systems have been proposed to detect road congestion using video [7] or image [8] processing technologies. However, these approaches require sensors and other equipment such as cameras and thus the deployment and maintenance are costly.

Several approaches have been proposed, particularly during the last decade, to use vehicular ad hoc networks (VANETs) for monitoring traffic [6, 9, 10], in general, and for specific purposes, such as for traffic coordination and disaster management [11–13]. Simulations have also been playing a key role in transportation planning and control [14]. A number of works on operations research related to transportation in smart cities have also been proposed, e.g., car-free cities [15], intelligent mobility [16], big data in transport operations [17, 18], prototyping in urban logistics [19], and autonomic transportation systems [20–22]. Furthermore, Alomar et al. [23] visualized traffic incidents in the city of Riyadh for the 2013–2015 timeframe. However, they did not work on social data. They get the data from the General Directorate of Traffic (GDT). Other researchers study road traffic crashes in Pakistan during Ramadan [24]. They also analyzed structured data from formal sources.

2.2.2 Event Detection from Social Media

Several approaches have been proposed to detect events from social data in different languages.

Kurniawan et al. [25] conducted experiments to classify real-time road traffic tweets using data mining. They collected real-time data about Yogyakarta Province, Indonesia using Twitter Streaming API. Additionally, they compared classification performance of three machine learning algorithms, namely Naive Bayes (NB), Support Vector Machine (SVM), and Decision Tree (DT). However, they only classified tweets into the traffic or non-traffic categories. Similar work is proposed by D'Andrea et al. [26]. They suggested an intelligent system, based on text mining and machine learning algorithms. They collected real-time tweets of several regions of the Italian road networks and then assigned the appropriate class label to each tweet, as to whether the tweet is related to a traffic event or not.

Ribeiro et al. [27] analyzed tweets to detect traffic events in Belo Horizonte, Brazil. They created a set of place names, called GEODICT. Subsequently, they detected the locations and streets names by using string matching technique by searching for substrings from the tweet that can be detected in GEODICT. Wongcharoen and Senivongse [28] built a congestion severity prediction model to

predict traffic congestion severity level. However, like previous approaches [25, 27], the tweets are fetched only from particular accounts.

Hanifah et al. [29] filtered tweets using SVM to detect traffic congestion in Bandung, Indonesia. Also, they extracted the information of location, time, date, and image. For information extraction, they applied a rule-based approach, which is based on the structure of the sentence. However, they did not detect traffic-related events. Gu et al. addressed this limitation [30]. They have collected historical and real-time tweets about traffic in Pittsburgh and Philadelphia, Metropolitan. They used a dictionary of relevant keywords and their combinations that can indicate traffic condition.

Moreover, D'Andrea et al. [26] collected real-time Italian tweets and classified them into traffic and non-traffic tweets. Alifi and Supangkat [31] suggested approaches for extracting location information. Additionally, they extracted valuable information from real time including traffic condition, congestion causes, weather condition, and time of occurrence. However, researchers in [26, 29, 30], and [31] did not perform sentiment analysis. Additionally, none of them applied big data technologies in their proposed methods. Suma et al. [32, 33] have analyzed Twitter data to detect events related to road traffic and other topics for smart cities planning purposes. Their focus is on the use of big data platforms to analyze large amounts of tweets about the London city. However, they did not perform sentiment analysis. Moreover, in our previous work [5] we used SAP HANA to detect road traffic conditions in Jeddah city. However, we did not perform SA.

Several approaches have been proposed to detect events from Arabic social data. AL-Smadi and Qawasmeh [34] used an unsupervised rule-based technique to extract events about technology, politics, etc. In [35], the researchers detect events related to disasters, sports, arts, crime, politics, and elections. Other researchers classified real-time tweets to detect high-risk floods [36]. Moreover, researchers in [37] annotated Arabic events related to politics and election. Furthermore, Alsaedi and Pete [38] proposed a framework for detecting disruptive events from Arabic tweets. They extended their work and suggested an integrated event detection framework related to the riots events [39]. However, none of these studies focused on traffic events.

2.2.3 Arabic Sentiment Analysis

The existing work about Arabic sentiment analysis (not specific to transportation) can be classified into lexicon (dictionary) based, ML-based, or hybrid. Researchers in [40–42] applied a hybrid approach for Jordanian dialect. On the other side, there are some studies based on machine learning for Modern Standard Arabic (MSA) [43], Egyptian dialect [44], and Jordan dialect [45]. Furthermore, researchers in [46, 47] proposed lexicon-based Arabic SA, but they are not proposed for Saudi dialect.

Few studies have applied SA to Saudi dialect. Aldayel and Azmi proposed hybrid (SVM and lexical) classifier [2]. However, they only performed two-way (positive, negative) classification. Moreover, the Saudi dialect lexicon has been developed

in [48]. But, it is domain specific (restaurants reviews). Al-twairesh proposed AraSenTi-tweet [49] corpus for sentiment analysis. It is available online for the research community. Even though the corpus annotated manually, they extracted from a large dataset that contains Arabic tweets. Most of the existing words in their lexicon are not useful in our case (traffic detection). Further, some of them do not belong to the Saudi dialect.

From the above discussion for the literature review, we found that the existing Arabic sentiment lexicons are either not supporting Saudi dialect or not efficient to be used in traffic detection domain. Therefore, there is a need to create a new sentiment lexicon to classify the traffic-related tweets.

On the other side, big data processing technologies provide great opportunities for addressing transportation problems for which traditional approaches are not competent. To the best of our knowledge, none of the existing work about event detection from Arabic social data has used big data platforms and technologies to address the complex processing and analytics tasks on such big data. Therefore, our text classification technique will be built on SAP HANA, which is an in-memory processing platform offering groundbreaking performance.

2.3 Methodology

Figure 2.1 illustrates the workflow of tweets acquisition, processing, and analytics. We have built our approach on SAP HANA, which is developed by SAP SE. It is the integration of transactional and analytical workload within the same database

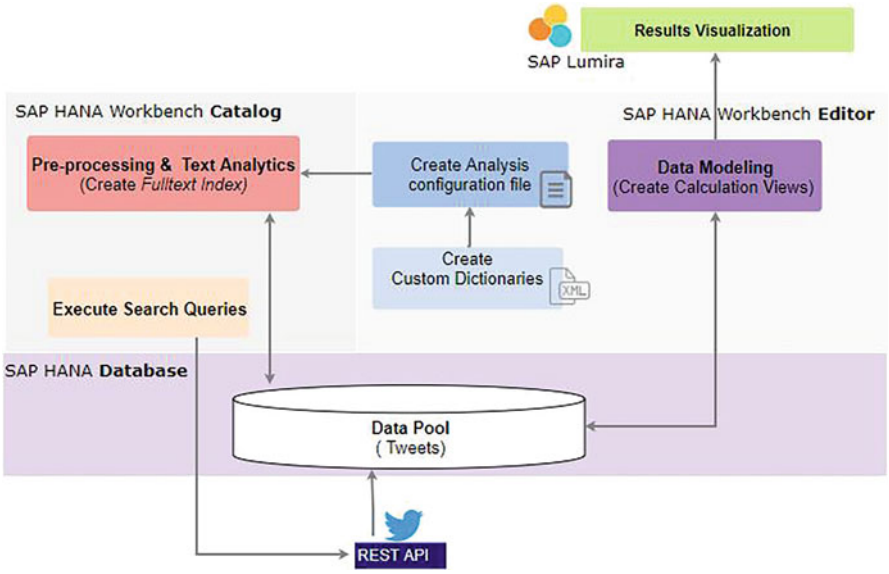


Fig. 2.1 Overview of the main implementation steps

management system [50]. Further, SAP HANA Extended Application Services (SAP HANA XS) provides the SAP HANA *Web-based Development Workbench* that supports developing entire applications in a Web browser without the need to install any development tools. SAP HANA Web-based Development Workbench includes i) Catalog and ii) Editor tools [51].

Catalog enables developing and maintaining SQL catalog objects in the SAP HANA database. It also supports creating tables, executing SQL queries, and creating a remote source to collect data. Additionally, *catalog* supports text analysis and text mining. Moreover, *Editor* enables data modeling, which is an activity of creating information view. This information views can be used for reporting and decision-making purpose. SAP HANA supports a great information view, which is a calculation view. The data foundation of the calculation view can include tables, column views, analytic views, and calculation views. Also, it enables creating joins, unions, aggregation, and projections on data sources.

2.3.1 Tweets Collection

We collected tweets about traffic in Jeddah and Makkah during Ramadan (17 May–14 June), 2018. We generated a list of Arabic keywords related to road traffic and transportation. We also searched for the most popular Twitter accounts that tweet about traffic conditions in Jeddah and Makkah cities. We have used the collected list of twitter accounts and Arabic keywords to write a large number of queries.

Search queries were executed in SAP HANA Workbench *Catalog* to collect historical tweets using twitter REST search API. Unlike streaming API that enables fetching real-time tweets, the REST API allows us to query historical tweets with locations and keywords simultaneously. REST API supports geocode parameter to restrict query by a given location using "latitude, longitude, radius." Thus, when executing the queries, the search API will first attempt to search for tweets which have lat/long within the queried geocode. But not all tweets are geotagged because some users disable location service in their smartphones. In this case, Tweet's location information will be detected from the location data in the user's profile.

However, if the user did not add information about the city and county in his/her profile, "Country" and "Place_name" fields would be empty. To handle this issue and fetch the non-geotagged tweets, we re-execute all queries after adding the city name and without specifying a location to collect all traffic tweets that include the city name. However, there are still some tweets that are not included in our analysis because they are not geotagged and not carrying location information. We created a table to store the retrieved tweets in SAP HANA databases. The created table includes several attributes such as "UserId," "Tweet," "UserName," "CreatedAt," "Latitude," "Longitude," "Country," and "Place_name."

2.3.2 *Pre-processing and Analysis Configuration*

SAP HANA supports text analysis for different languages including Arabic. They used the pre-processor server to extract and classify unstructured text into entities and domains by applying linguistic and statistical techniques [52]. To analyze the text in SAP HANA, there is a need to create full-text indexing on the text column with specifying the type of analysis configuration and setting TEXT ANALYSIS parameter "ON" and this results in a new table "$TA__<index name>". This table will include linguistic or semantic analysis results.

SAP HANA supports three main types of text analysis configurations, which are [53]:

- Linguistic Analysis: supports natural language processing.
- Entity and Fact Extraction: enables named entity extraction, sentiment analysis, public sector events, and enterprise facts. It named EXTRACTION_CORE_VOICEOFCUSTOMER.
- Grammatical Role Analysis: enables functional syntactic roles in the sentence, such as subject or object. It supports English language only.

In this work, the data are analyzed based on "Voice Of Customer" (VOC) analysis configuration. We have selected this type of text analysis configuration because it supports handling entity extraction, fact extraction, and sentiment analysis. Further, it enables tokenization, which means it decomposes the phrase or sentence into tokens. Unlike "Linguistic analysis" configuration that extracts every word in the text, VOC extracts only basic entities from the text and entities of interest including a person, address, organization, URLs, and other common terms. The token type is stored in TA_TYPE field.

To use the default configuration, developers simply need to include VOICEOFCUSTOMER parameter in a query. However, the standard configuration doesn't suffice to the requirement especially with the Arabic language. Further, the default normalizer is not efficient. Thus, we need to customize keywords in new dictionaries and include them in a modified configuration file.

Custom Dictionaries

We noticed that the standard text analysis in SAP HANA using the VOICEOF CUSTOMER-configuration does not suffice where not all Arabic tokens are classified under the right token type. Therefore, we need to add a custom dictionary for unknown terms in the SAP HANA system and then create a new configuration file. We created our own dictionaries because none of the existing dictionaries for Saudi dialect are designed to be used for road traffic condition detection. The created lists of custom dictionaries were used to create a new configuration file for analysis using SAP HANA Web-based Development Workbench. Then, the generated configuration file was used to create the fulltext index on "Tweets"

column to split the text into tokens and specify the token type based on the created dictionaries.

We created several custom dictionaries, which help to improve tokenization, normalization, and entity type extraction. The main dictionaries are as follows:

- Transportation: includes the collected Arabic keywords about transportation (such as سيارة, ساهر, إشارة).
- Makkah Streets/Jeddah Streets: contain the names of streets and roads names.
- Places: includes the keywords referred to places names like Mosque, Restaurant, and Mall.
- Religion: contains the synonyms of words related to fasting and the activities during Ramadan month (e.g., افطار, عمره, التراويح).
- Sentiment: includes a list of Arabic and Saudi dialect sentiment words and expression.
- Events types: contains the common words representing events types and list of their corresponding synonyms.

Tokenization, Normalization, and Entity Extraction.

To analyze the tweets in SAP HANA, we need to create a full-text index on "Tweet" column. Creating the index requires executing SQL statement, which will lead to creating a new table containing the tokens and named entity extraction results. The created table will include the following:

- TA_Token: contains the list of tokens extracted from the tweets.
- TA_Type: refers to the entity type.
- TA_Normalized: stores a normalized representation of the token.

The created custom dictionaries enable identifying a standard name for each entity. The TA_Type field can contain built-in type (e.g., NOUN_GROUP) or one of the types that are specified in our newly created dictionaries, i.e., Jeddah_Street. Moreover, the normalization process is very important especially for Arabic text where some letter has different representation. For instance, "Alif" has four forms (أ, ا, إ, آ), "Yaa" has two forms (ي, ى), and "Haa" has two forms (ه, ة). SAP HANA supports case normalization by converting the initial letter of a word to upper or lower case. However, this type of normalization is not relevant to languages that do not distinguish between upper and lower case such as Arabic. So, we modified the analysis configuration to represent the normalized form of the entity as specified in our custom analysis dictionaries. For example, "زحمة" and "المدينة" will be normalized to "زحمه" and "المدينه" where "TAA MARBUTAH/ة" was replaced with "HAA/ه."

2.3.3 Tweets Analysis

Location Extraction

Generally, there are two types of location information: (i) Latitude/longitude coordinates of the locations where users posted the tweets and (ii) Location name referred in tweet texts. We specified either coordination information or cities name in our search queries to force them to retrieve only tweets posted in our targeted cities. Further, to extract specific location information such as streets name from the text, we used the Entity Extraction feature in SAP HANA. However, the existing entity extractor with default configuration did not detect all the places names. So, we created our own dictionaries for the main streets/roads names and then we included them in the modified configuration file. We used OpenStreetMap[1] to create a list of streets and roads names in Jeddah and Makkah. When we run the analysis query (create full-text index), the places name will be extracted from the text and stored in the analysis results table.

Traffic Events Detection

We created a dictionary containing a list of words representing the road traffic events. We took into account the following events:

- Accident (حادث).
- Fire (حريق).
- Roadworks "أعمال الطرق" including maintenance (صيانة) and construction (بناء).
- Weather condition "الطقس" such as rain (مطر) and storm (عاصفة).
- Other events that could affect the traffic including sports (رياضة) events and social events (e.g., festival "مهرجان").

We expand the dictionary by adding a list of corresponding synonyms under each event type. Consequently, each type of traffic event is extracted taking into account the set of relevant words. For instance, accident "حادث" associated with words like "صدم," and maintenance "صيانه" associated with words like "ترميم" or "اصلاح." To clarify, during the tokenization and entity extraction phases, each token will get a Token_Type based on our custom dictionaries where our event detection technique relies on matching synonyms with terms available on the tweet. For instance, the following tweet contains the word "حريق," and thus the extracted event type will be "fire." We consider the fires as traffic-related events even though it is not a vehicle fire because it may effect on the traffic condition and cause congestion.

Example: "@JeddahNow: حريق ضخم في متاجر #اكسترا على شارع التحلية ، مع تواجد كثيف لفرق الدفاع المدني ، سنوافيك بالمستجدات لاحقاً مباشر #جدة."

[1] http://openstreetmap.org

Translation: "@JeddahNow: Live #Jeddah | A huge fire at # Extra stores on Tahlia Street, with an intensive presence of the Civil Defense teams, we will update you about the status soon."

Sentiment Analysis

The literature review suggests two approaches for building a lexicon: manual construction by experts or automatic construction. Although automatic lexicon construction from a seed of words is faster and required less human effort, there are weaknesses regarding accuracy and robustness due to the lack of human supervision. Thus, in this work, we followed a lexicon-based approach that relies on a manually constructed dictionary. We built lexicons for Saudi dialect words that related to traffic condition. We created a list of strong positive words (e.g., اسرع "Faster"), positive words (e.g., سالك "no traffic jam"), negative words (e.g., بطيء "Slow"), and strong negative words (e.g., وفاه "Death"). Then, we expanded the lists by adding synonyms.

After that, we included the created custom dictionaries in the analysis configuration file. When we created a full-text index, the analyzer simply splits each word in the tweet, normalize it using our dictionaries, then classify each token in the tweet into one of the four categorized. Subsequently, we created a calculation view to classify the tweets. Each tweet will be scored based on the number of the tokens from each sentiment class and on how many times these words occurred in the text. Subsequently, the tweets are classified appropriately based on the calculated score.

2.4 Results and Dissection

SAP offers a data visualization tool for reporting on top of SAP HANA, named SAP Lumira.[2] Figure 2.2 shows the percentage of tweets at different time of day. The chart in Fig. 2.2a shows that most tweets about traffic in Jeddah are posted during the night. The highest tweeting time is at 22. The percentage of tweets is started decreasing after 3 and the lowest tweeting time is at 8. The results are reasonable where the business hours during Ramadan are changed, and people used to go to the markets and restaurants before Iftar in addition to that they usually go shopping after Al-Taraweeh prayer Additionally, during Ramadan, the work hours are changed, and most employees in public sector and private companies work from 10 am to 3 pm.

On the other side, Fig. 2.2b shows that the percentage of tweets about traffic in Makkah is always high except for the period between Al-Fajr prayer and Al_Dhuhr

[2] http://saplumira.com/

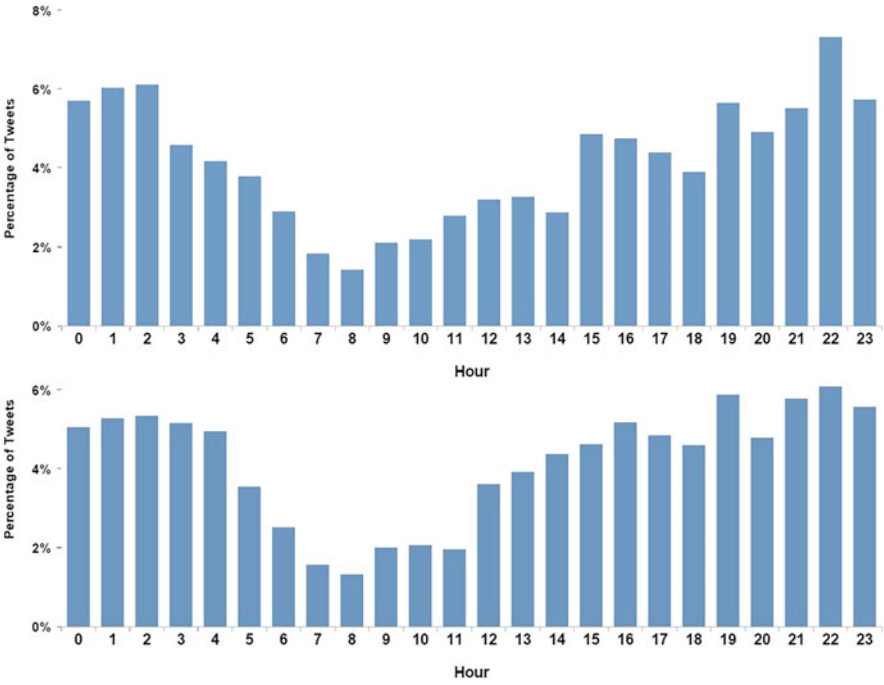


Fig. 2.2 Percentage of Tweets on different time. (**a**) Jeddah. (**b**) Makkah

prayers (5–12). Like the tweets about Jeddah, the number starts decreasing after Al-Fajr prayer where most people used to sleep at this time.

Moreover, we filtered the collected tweets to show only the tweets messages that contain street/road names. After that, we drew a chart to represent the top mentioned street/road. However, we noticed that the number of tweets messages that contains place name is not very large. The main reasons that could explain that are (i) the limit in the characters number in Twitter, (ii) people may post a message to reply to another tweets or participate in a hashtag about specific events, which don't required re-mentioning the name of the place, (iii) the tweets that describe feelings or emotions usually do not contain a specific place name.

As shown in Fig.2.3a, the most mentioned names in the collected tweets about traffic in Jeddah are Prince Sultan St., Altahliah St., King Abdul Aziz Rd., Palatine St., and Almadinah Rd. On the other side, Fig. 2.3b illustrates that the top five mentioned roads/streets names in the tweets about Makkah, which are Makkah-Jeddah highway, Alhaj street, Almadinah Almunawwarah road, Ajyad street, and Alsail road. This result is reasonable where millions of Muslims visited Makkah in Ramadan to perform Umrah and pray in Al-Masjid Al-Haram, which could affect the traffic to/from the city, in addition to the traffic to/from Al-Haram. Ajyad is one

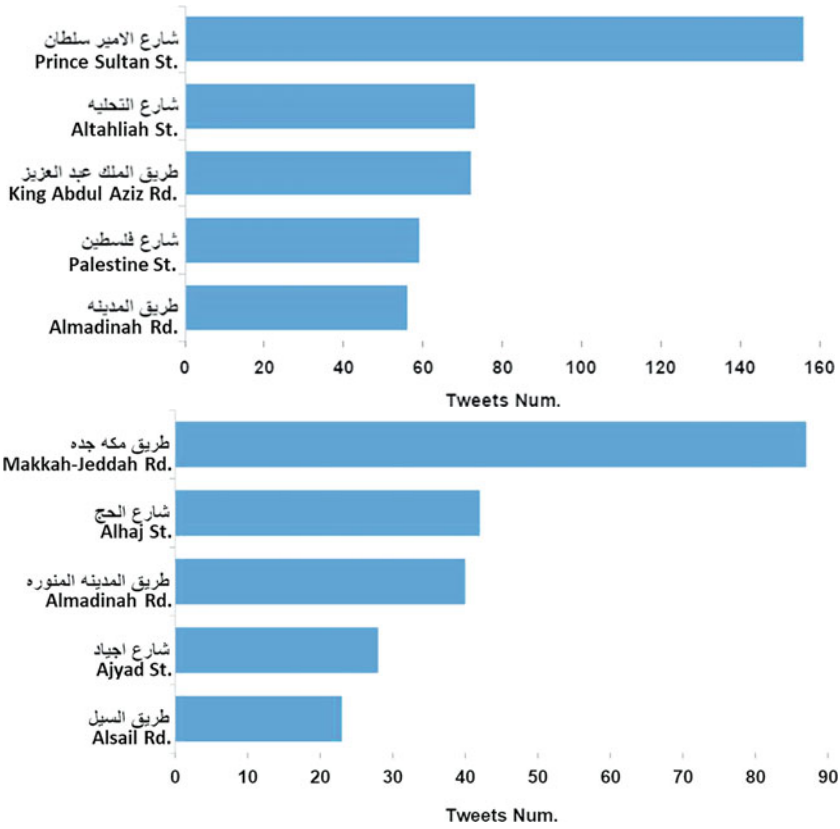


Fig. 2.3 Top mentioned roads/street names. (**a**) Jeddah. (**b**) Makkah

of the main streets leading to Alharam. Alhaj street is one of the main streets in Makkah and connects many districts. The other roads are the main roads connecting Makkah with Jeddah, Al-Madinah, and Al-Ta'if (Alsail Rd.) cities.

Furthermore, Fig. 2.4 illustrates the top detected events in Jeddah and Makkah. The events are detected based on the existing of terms in the created dictionaries. In this work, we exclude the retweets (repost of another user's posts) except when detecting the top mentioned events. The number of retweets is an indication of popularity. Further, it has been implemented to detect events [38]. So, we included the retweets number when detecting the top events.

As shown in Fig. 2.4a, the top three detected events in Jeddah are accidents, fires, and inauguration. To validate our event detection mechanism, we searched in newspapers websites (Okaz**,** Sabq, etc.) to compare the results. We found that there was a fire in "Extra Store" (on May 28) near Altahliah St., and another building

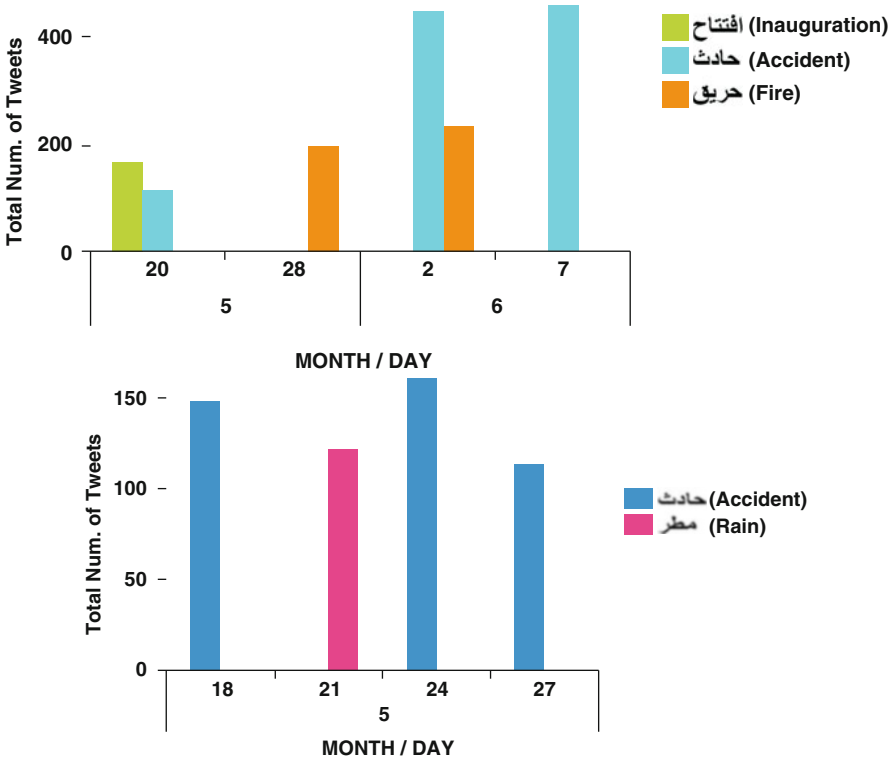


Fig. 2.4 Top detected events. (**a**) Jeddah. (**b**) Makkah

fire (on June 2) leads to 14 injured. In addition, Jeddah Municipality announced that construction work in Al-Andalus Tunnel was done and the tunnel inaugurated at the beginning of Ramadan. We also found articles about several car accidents occurred during Ramadan, one of them was on June 2, where a driver lost control of his car. Further, our tool detected accident on June 7. However, we discovered from searching that the accident occurred on June 5, but there were new posts about it two days later when a nurse honored by the ministry of health for helping injured people in that accident while she is out of work hours.

Moreover, Fig. 2.4b shows the top detected events in Makkah which are rains and accident. We found posts on online newspapers about rains in Makkah on May 21. Additionally, our tool detected several accidents during Ramadan. One of them was on May18. We found details in newspapers articles where there were 9 deaths and 18 injured in a bus accident. Additionally, we found posts about another car accidents (on May 24) in the road connects between Makkah and Al-Madinah cities. From the above discussion, we can notice that the developed tool can automatically detect the traffic events from twitter posts.

Table 2.1 Examples of sentiment classification for driver's feeling and opinions

	Tweets	English Translation	Classification
1	من هذا الشارع بالذات الزحام غير طبيعي	From this street, the congestion is unusual	Negative
2	امطار مكه المكرمة صورة احد شوارع العاصمة المقدسة روعه	Rains in Makkah Almukarramah, a photo of one of the streets in the holy capital, magnificence.	Positive
3	لا يعقل مشروع نفق طريق الأندلس تقاطع شارع فلسطين قبل يومين افتتاحه واليوم صيانه ومسار واحد فقط مما سبب زحمة .. للأسف...	Unbelievable, the tunnel project of Al-Andalus Rd. Palestine street intersection opened two days ago, and today maintenance, only one-way opened, which caused a traffic jam ... Unfortunately ...	Strong negative
4	@jedgovsa نفق او كبري او طريق الاندلس بجدة ممتاز وصرحة ابداع الي افتتحتوه قبل كم يوم او با الكثير شهر	@jedgovsa Alandalus tunnel or bridge or road in Jeddah is excellent and creative, opened a few days or a month ago.	Strong positive

Table 2.1 shows examples of sentiment classification for driver's feelings and opinions. We gave an English translation for non-Arab readers. We provided a literal translation to avoid giving meaning from our side. The tweets are classified into one of 4 sentiment classes based on the total score that is calculated after dividing the text into tokens and identifying the class for each token. For instance, the combination of the two negative terms "congestion" and "unusual" in tweet#1 leads to classifying the tweet as negative. Furthermore, the word "Rain" is labeled as negative where it almost causes negative effect on traffic. However, the existence of the word "magnificence" in tweet#2, which is a strong positive keyword leads to classifying the tweet as positive.

Furthermore, we draw a chart to illustrate the list of the top mentioned words related to the causes of congestions. Figure 2.5 indicates that the word "حادث" (accident) was the most traffic cause mentioned in the collected tweets about traffic in Makkah and Jeddah. Figure 2.6 shows the word cloud for the top used terms about roads and traffic which include street "شارع," road "طريق," accident "حادث," and congestion "زحام."

2.5 Conclusions

In this work, we analyzed Saudi dialect tweets about road traffic conditions. We collected tweets during Ramadan and focused on two large cities (Jeddah and Makkah). We developed our method on SAP HANA, which is an in-memory processing platform to store and analyze the data. The default analysis configuration in SAP HANA is not efficient for Arabic text analysis. So, we created a new configuration file. We added new dictionaries for the Arabic and Saudi dialect

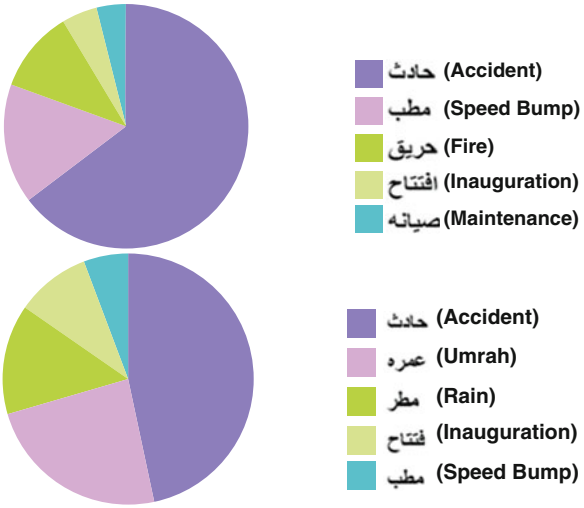


Fig. 2.5 Top mentioned terms related to congestion causes. (**a**) Jeddah. (**b**) Makkah

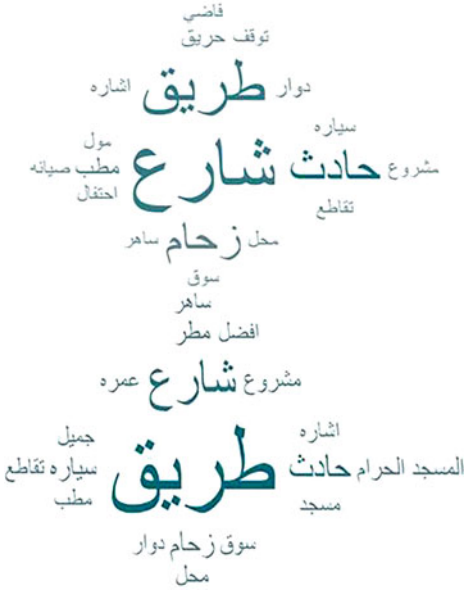


Fig. 2.6 The most frequent terms. (**a**) Jeddah. (**b**) Makkah

keywords related to sentiment, traffic events, and streets names. These dictionaries help in improving tokenization, normalization, and entity extraction. The main contributions of this work are detecting traffic-related events and applying sentiment analysis based on lexicon approach to classify driver's feeling and emotions.

Moreover, we have used SAP Lumira to visualize the results by creating charts. We drew a chart to represent the top mentioned traffic events in the tweets. Additionally, we showed the most frequently mentioned terms related to congestion causes. To validate the proposed event detecting mechanism, we compared the results with data from local newspapers websites. In the future, we plan to measure the accuracy of our proposed sentiment classification approach. Additionally, we will expand our sentiment lexicon and include more words.

References

1. Wang, S., He, L., Stenneth, L., Yu, P.S., Li, Z.: Citywide Traffic Congestion Estimation with Social Media
2. Aldayel, H.K., Azmi, A.M.: Arabic tweets sentiment analysis – a hybrid scheme. J. Inf. Sci. **42**(6), 782–797 (2016)
3. Mourtada, R., Salem, F., Al-Shaer, S.: Citizen engagement and public services in the Arab world: the potential of social media. *Arab Soc. Media Rep.*, no. 2014
4. www.statista.com, Twitter: number of active users 2010–2016 | Statista. 2016
5. Alomari, E., Mehmood, R.: Analysis of Tweets in Arabic Language for Detection of Road Traffic Conditions, pp. 98–110. Springer, Cham (2018)
6. Mehmood, R., Nekovee, M.: Vehicular AD HOC and grid networks: discussion, design and evaluation. In: 14th World Congress on Intelligent Transport Systems, ITS 2007, vol. 2, pp. 1555–1562 (2007)
7. Kanungo, A., Sharma, A., Singla, C.: Smart traffic lights switching and traffic density calculation using video processing. In: 2014 Recent Advances in Engineering and Computational Sciences (RAECS), pp. 1–6 (2014)
8. Wei, L., Dai, H.-Y.: Real-time road congestion detection based on image texture analysis. Procedia Eng. **137**, 196–201 (2016)
9. Gillani, S., Shahzad, F., Qayyum, A., Mehmood, R.: A survey on security in vehicular ad hoc networks, vol. 7865 LNCS. (2013)
10. Alvi, A., Nabi, Z., Greaves, D.J., Mehmood, R.: Intra-vehicular verification and control: a two-pronged approach. Int. J. Veh. Inf. Commun. Syst. **2**(3–4), 248–268 (2011)
11. Alazawi, Z., Altowaijri, S., Mehmood, R., Abdljabar, M.B.: Intelligent disaster management system based on cloud-enabled vehicular networks, in 2011 11th International Conference on ITS Telecommunications, ITST 2011, pp. 361–368 (2011)
12. Alazawi, Z., Abdljabar, M.B., Altowaijri, S., Vegni, A.M., Mehmood, R.: ICDMS: An intelligent cloud based disaster management system for vehicular networks, vol. 7266. Springer, Vilnius, Lithuania (2012)
13. Alazawi, Z., Alani, O., Abdljabar, M.B., Altowaijri, S., Mehmood, R.: A smart disaster management system for future cities, WiMobCity '14. Int. Work. Wirel. Mob. Technol. Smart Cities, pp. 1–10, (2014)
14. Ayres, G., Mehmood, R.: On discovering road traffic information using virtual reality simulations, in 11th International Conference on Computer Modelling and Simulation, UKSim 2009, pp. 411–416 (2009)
15. Mehmood, R., Lu, J.A.: Computational Markovian analysis of large systems. J. Manuf. Technol. Manag. **22**(6), 804–817 (2011)
16. Büscher, M., Coulton, P., Efstratiou, C., Gellersen, H., Hemment, D., Mehmood, R., Sangiorgi, D.: Intelligent mobility systems: some socio-technical challenges and opportunities. In: Mehmood, R., Cerqueira, E., Piesiewicz, R., Chlamtac, I. (eds.) Communications Infrastructure. Systems and Applications in Europe, pp. 140–152. Springer, Berlin (2009)

17. Mehmood, R., Meriton, R., Graham, G., Hennelly, P., Kumar, M.: Exploring the influence of big data on city transport operations: a Markovian approach. Int. J. Oper. Prod. Manag. **37**(1), 75–104 (Jan. 2017)
18. Mehmood, R., Graham, G.: Big data logistics: a health-care transport capacity sharing model. Procedia Comput. Sci. **64**, 1107–1114 (2015)
19. Graham, G., Mehmood, R., Coles, E.: Exploring future cityscapes through urban logistics prototyping: a technical viewpoint. Supply Chain Manag. **20**(3), 341–352 (2015)
20. Schlingensiepen, J., Mehmood, R., Nemtanu, F.C.: Framework for an autonomic transport system in smart cities. Cybern. Inf. Technol. **15**(5), 50–62 (2015)
21. Schlingensiepen, J., Mehmood, R., Nemtanu, F.C., Niculescu, M.: Increasing sustainability of road transport in European Cities and metropolitan areas by Facilitating Autonomic Road Transport Systems (ARTS). In Sustainable Automotive Technologies 2013 Proceedings of the 5th International Conference ICSAT 2013, pp. 201–210 (2014)
22. Schlingensiepen, J., Nemtanu, F.: Autonomic transport management systems—enabler for smart cities, personalized medicine, participation and industry grid/industry 4.0. In: Sladkowski, A., Pamula, W. (eds.) Intelligent Transportation Systems – Problems and Perspectives, pp. 3–35. Springer International Publishing, London (2016)
23. H.A., Alomar, A., Alrashed, N., Alturaiki, I.: How visual analytics unlock insights into traffic incidents in urban areas. In: Business (2017)
24. Mehmood, A., Khan, I.Q., Mir, M.U., Moin, A., Jooma, R.: Vulnerable road users are at greater risk during ramadan—results from road traffic surveillance data. J. Pak. Med. Assoc. **65**(3), 287–291 (2015)
25. D. A. Kurniawan, S. Wibirama, and N. A. Setiawan, Real-time traffic classification with Twitter data mining, 2016
26. D'Andrea, E., Ducange, P., Lazzerini, B., Marcelloni, F.: Real-time detection of traffic from twitter stream analysis. IEEE Trans. Intell. Transp. Syst. **16**(4), 2269–2283 (2015)
27. Ribeiro, S.S., Davis, C.A., Oliveira, D.R.R., Meira, W., Gonçalves, T.S., Pappa, G.L.: Traffic observatory: a system to detect and locate traffic events and conditions using Twitter Sílvio. Proc. 5th Int. Work. Locat. Soc. Networks—LBSN '12, p. 5, (2012)
28. Wongcharoen, S., Senivongse, T.: Twitter analysis of road traffic congestion severity estimation. In 13th Int. Jt. Conf. Comput. Sci. Softw. Eng. (2016)
29. Hanifah, R., Supangkat, S.H., Purwarianti, A.: Twitter information extraction for smart city. In Proc.—2014 Int. Conf. ICT Smart Soc. Smart Syst. Platf. Dev. City Soc. GoeSmart 2014, ICISS 2014, pp. 295–299, (2014)
30. Gu, Y., (Sean) Qian, Z., Chen, F.: From twitter to detector: real-time traffic incident detection using social media data. Transp. Res. Part C Emerg. Technol. **67**, 321–342 (2016)
31. Alifi, M.R., Supangkat, S.H.: Information extraction for traffic congestion in social network. In International Conference on ICT For Smart Society, pp. 20–21 (2016)
32. Suma, S., Mehmood, R., Albugami, N., Katib, I., Albeshri, A.: Enabling next generation logistics and planning for smarter societies. Procedia—Procedia Comput. Sci., pp. 1–6. (2017)
33. Suma, S., Mehmood, R., Albeshri, A.: Automatic event detection in smart cities using big data analytics. In International Conference on Smart Cities, Infrastructure, Technologies and Applications SCITA 2017: Smart societies, Infrastructure, Technologies and Applications, pp. 111–122 (2018)
34. AL-Smadi, M., Qawasmeh, O.: Knowledge-based approach for event extraction from Arabic Tweets. Int. J. Adv. Comput. Sci. Appl. **7**(6), (2016)
35. Hasanain, M., Suwaileh, R., Kutlu, T.M., Elsayed, H.A.: EveTAR: building a large-scale multi-task test collection over Arabic Tweets, arXiv Prepr. arXiv1708.05517., (2017)
36. Alabbas, W., Haider, M., Mansour, A., Epiphaniou, G., Frommholz, I.: Classification of colloquial Arabic Tweets in real- time to detect high-risk floods. Soc. Media, Wearable Web Anal. (Social Media), 2017 Int. Conf. IEEE., 2017
37. Aliane, H., Information, T., Guendouzi, A., Mokrani, A.: Annotating events, time and place expressions in Arabic texts. In Proceedings of Recent Advances in Natural Language Processing, pp. 25–31. (2013)

38. Alsaedi, N., Burnap, P.: Arabic event detection in social media. In LNCS, vol. 9041, pp. 384–401 (2015)
39. Alsaedi, N., Burnap, P., Rana, O.: Can we predict a riot? Disruptive event detection using Twitter, vol. 17, no. 2, (2017)
40. Siddiqui, S., Monem, A.A., Shaalan, K.: Towards improving sentiment analysis in Arabic. In Advances in Intelligent Systems and Computing, vol. 533, pp. 114–123 (2017)
41. Duwairi, S.R.R.M., Marji, R., Sha'ban, N.: Sentiment analysis in Arabic Tweets. In Information and communication systems (icics), 2014 5th international conference on. IEEE, vol. 12, no. 11 (2014)
42. Duwairi, R.M.: Sentiment analysis for dialectical Arabic. In 2015 6th International Conference on Information and Communication Systems, ICICS 2015, pp. 166–170 (2015)
43. Abdul-Mageed, M., Diab, M., Kübler, S.: SAMAR: Subjectivity and sentiment analysis for Arabic social media. Comput. Speech Lang. **28**(1), (2014)
44. Rafea, A., Shoukry, A., Rafea, A.: Sentence-Level Arabic sentiment analysis sentence-level Arabic sentiment analysis. In Collaboration Technologies and Systems (CTS), 2012 International Conference on. IEEE, (2012)
45. Alomari, K.M., Elsherif, H.M., Shaalan, K.: Arabic Tweets sentimental analysis using machine learning. In International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, pp. 602–610 (2017)
46. Abdulla, N.A., Ahmed, N.A., Shehab, M.A., Al-Ayyoub, M., Al-Kabi, M.N., Al-rifai, S.: Towards improving the lexicon-based approach for Arabic sentiment analysis. Int. J. Inf. Technol. Web Eng. **9**(3), 55–71 (2014)
47. Al-Horaibi, L., Khan, M.B.: Sentiment analysis of Arabic Tweets using semantic resources. Int. J. Comput. Inf. Sci. **12**(2), (2016)
48. Al-Hussaini, H., Al-Dossari, H.: A Lexicon-based approach to build service provider reputation from Arabic Tweets in Twitter, (IJACSA). Int. J. Adv. Comput. Sci. Appl. **8**(4), (2017)
49. Al-twairesh, N., Al-khalifa, H., Al-salman, A., Al-ohali, Y.: AraSenTi-tweet: a Corpus for Arabic sentiment analysis of Saudi tweets. Procedia Comput. Sci. **117**, 63–72 (2017)
50. SAP, What is SAP HANA | In Memory Computing and Real Time Analytics, 2016.
51. SAP HANA Web-Based Development Workbench - Introduction to SAP HANA Development - SAP Library.
52. SAP HANA Text Analysis Language Reference Guide, 2016
53. SAP HANA Text Analysis Developer Guide, 2016

Chapter 3
Automatic Detection and Validation of Smart City Events Using HPC and Apache Spark Platforms

Sugimiyanto Suma, Rashid Mehmood, and Aiiad Albeshri

3.1 Introduction

Smart city developments are driving an unprecedented growth and innovation in everyday life, urban, rural, and elsewhere. Big data, high performance computing (HPC), and machine learning technologies are playing a key role in supporting smart city and society systems and applications. There is a need to sense the cities and other environments at micro-levels, to make intelligent decisions, and to take appropriate actions, all within stringent time bounds. Social media have revolutionized our societies and are gradually becoming a key pulse of smart societies by sensing the information about the people and their spatio-temporal experiences around the living spaces. All these developments together are driving the growth of data generation, i.e., the big data dimension. The management of various devices, sensors, and other entities, as well as the data generated by these entities, is an essential requirement. HPC and machine learning, together with big data are enabling the data management and the development of smart infrastructure to support smart cities and societies.

One of the key functions in smart cities and societies is the automatic detection of interesting events. Detecting an event is important in finding out what is happening

S. Suma (✉)
Division of Data, Department of Engineering, Kumparan, Jakarta Selatan, Indonesia
e-mail: sugimiyanto.sugimiyanto@kumparan.com

R. Mehmood
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: RMehmood@kau.edu.sa

A. Albeshri
Department of Computer Science, FCIT, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: aaalbeshri@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_3

in the city for decision-making or future planning purposes. Compared with sensor-based event detection, analyzing social media data such as twitter is a cost-effective way to detect events. Sensor-based detection analyzes traffic data collected from the installed sensors and cameras in certain places. It is costly and requires long-term planning due to the hardware procurement and network installation, among other things. In addition, the number of installed measurement instruments limits the detection coverage. Social media event detection has wider coverage, and is more efficient in terms of resources. However, they both have their pros and cons and could complement each other in terms of the convenience of event detection and information coverage.

In this paper, we continue our work on using twitter for the detection of spatio-temporal events in London. Specifically, we use big data, HPC, and AI platforms including Spark [1], and Tableau [2], to study twitter data about London. This paper extends our earlier work presented in [3, 4]. In [3], we had introduced our preliminary work on the use of social media for the detection of spatio-temporal events related to logistics and planning. In [4], we had improved on our data analytics architecture by implementing machine learning for contextual analysis awareness using Apache Spark MLlib. We use technologies that integrate big data and high performance computing (HPC) to improve performance of the event detection system. Big data and HPC convergence is an emerging area of research, see e.g., [5]. We use Apache Spark for parallel data processing, which is installed on top of the Aziz supercomputer [6]. We utilize the parallel file system FEFS system [7] for high-speed data distribution between the distributed Spark nodes. Moreover, we use the Google Maps Geocoding API [8] to locate the tweeters and make additional analysis.

We find and locate congestion around the London city. We also empirically demonstrate that events can be detected automatically by analyzing data. We detect the occurrence of multiple events including "Underbelly festival" [9] and "The Luna Cinema" [10]. Underbelly festival was located at south bank, while The Luna Cinema was located in multiple places including around Greenwich Park, Crystal Palace Park, and National Trust-Morden Hall Park. As well as, we detect the London Notting Hill Carnival 2017 event [11, 12]. This is located around Notting Hill as it was the location of Notting Hill carnival, the Europe's biggest street festival which was organized by London Notting Hill carnival enterprises trust. We detect those event's locations and times, without any prior knowledge of the event. The results presented in the paper have been obtained by analyzing over three million tweets. This paper makes the following specific enhancements over our earlier work [3, 4].

- Provides a comparison of three machine learning methods, support vector machine, logistic regression, and Naïve Bayes for event detection purposes using various performance metrics.
- Introduces an enhanced methodology to automatically validate the factuality of the detected events, i.e., to confirm that the events which were detected by our system did actually happen at the detected time and place.

- Provides an extended literature review.
- Elaborates on the methodology and architecture of the event detection and validation system and provides algorithms for the main components of the proposed system.

While researchers have studied social media based event detection in the recent past, the use of Apache Spark for social media based event detection has not been found in the literature. The specific data, its analysis, and event detection and validation proposed in our work also make our contributions novel.

The rest of the paper is organized as follows. Section 3.2 reviews the relevant literature. Section 3.3 introduces the design and methodology for the proposed event detection system. Section 3.4 provides a discussion on the results and analysis. Section 3.5 concludes the paper and gives future research directions.

3.2 Literature Review

Smart cities *"provide the state of the art approaches for urbanization, having evolved from the developments carried out under the umbrella of knowledge-based economy, and subsequently under the notion of digital economy and intelligent economy"* [13]. Smart society is an extension of the smart cities concept, "a digitally enabled, knowledge-based society, aware of and working towards social, environmental, and economic sustainability" [13]. Many new smart city applications are being developed, see e.g., knowledge learning and management [13], green computing [14], future applications [15], healthcare and life sciences [16–19], smart farming [20], disaster management [21], autonomous driving [22], and IoT-based smart applications [23].

Big Data refers to the *"emerging technologies that are designed to extract value from data having four Vs characteristics; volume, variety, velocity, and veracity"* [24]. Big data technologies are being used in many application areas, see e.g., [25–27].

Detecting events is an important area of research in many fields, such as in distributed systems [28] and eLearning [29]. Mobility and transportation (an important area of focus for event detection in this paper) is a key dimension of smart city designs and operations [30]. Many approaches have been proposed to address transportation challenges and develop smart transportation infrastructures, see e.g., autonomic transport systems [31–33], vehicular networks (VANETs) and systems [34–37], emergency management system [38–40], simulations [41, 42], urban logistics [25, 43, 44], big data [25–27], location-based services [45], and social media based approaches [3, 4].

Research for smart cities using big data analytics is becoming increasingly important. Rahman et al. proposed a forecasting system to predict the amount of power required by cities at a rate close to the electricity consumption in the United States [46]. Khan et al. developed a prototype analytics as a cloud service, for managing and analyzing big data in smart cities [47]. Herrera-Quintero et al.

combined big data and IoT to support transportation planning system for Bus Rapid Transit (BRT) systems [48]. Kolchyna et al. predicted spikes in sales by detecting twitter events of 150 million tweets [49]. In order to enable smarter cities with enhanced mobility information, Arfat et al. [50] proposed an architecture for smart city as a mobile computing system with big data technologies, fogs, and clouds.

Researchers have addressed data management and analysis in the past. Garcia et al. [51] reviewed data preprocessing methods, the definitions, categorization, and characteristics. They also discussed research challenges on developments on different big data frameworks such as Hadoop, Spark, and Flink. Fang & Zhan [52] proposed a general process for sentiment polarity categorization which aims to classify positive or negative users sentiment. Event detection cases were conducted using various methods. Hu et al. [53] proposed event detection techniques for social networks (interaction call and mail) according to link prediction. Ma et al. [54] proposed multimedia event detection approach which exploits the external concepts-based videos and event-based videos simultaneously. Rao et al. [55] developed a probabilistic detection of crowd events with various categories (running, walking, splitting, merging, and evacuation) by analyzing video data.

For spatio-temporal event detection purposes, exploiting social media data could complement traditional method using installed sensors and cameras. There have been a number of works analyzing social media data for detecting event. Doulamis et al. [56] proposed an approach for event detection in Twitter dataset using fuzzy technique. Events are detected through a multi-assignment graph partitioning algorithm and were performed on python system. Gu et al. [57] developed a real-time detector of traffic incident with five categories, including occurring events. It applies semi-Naïve Bayes classification. Nguyen and Jung [58] proposed an approach for early event identification, by combining content-based features from the social text data and the propagation of news between viewers. Unankard et al. [59] identified strong correlations between user location and event location to detect emerging hotspot events.

Wang [60] combined visual sensors (cameras) with social sensors (twitter feeds) to detect events. Image processing is applied to detect abnormal patterns indicating occurring events. Kaleel and Abhari [61] proposed an algorithm to detect interesting events by matching its keywords on cluster labels of tweet (clustering). Subsequently, trend is based on time, geo-locations, and cluster size. Tonon et al. [62] focused on detecting events related to natural disasters and terrorist activity using Twitter data. Pandhare et al. [63] have classified tweets to distinguish whether or not the tweets are related to traffic.

There are several other event detection works related to road traffic [64–66]; however, none of them use big data technologies and they use different analysis technique. While researchers have studied social media based event detection in the recent past, the use of Apache Spark for social media based event detection has not been properly investigated in the literature. The specific big data, focus detection, its analysis, and the event detection method and workflow presented in this paper also make the contributions of this paper unique. Note that this paper considers tweets in English language only. Another similar strand of our work considers event detection using tweets in Arabic language [67].

3.3 Methodology and Design

We developed a system architecture to detect spatio-temporal events as shown in Fig. 3.1. First, we crawl the status message (twitter) according to a predefined keyword set and a set of social media user accounts, which is relevant to traffic. Thereafter, we store the crawled data into a data pool. Secondly, we preprocess the acquired raw data before going to classification learning, where-upon social media data has lots of noises. It is not standardized, and there are plenty of unnecessary characters and words. Third, detecting events are using supervised learning for targeted event (traffic or non-traffic) and word frequency analysis for general events. Fourth, the detected general events are validated by retrieving information from the internet. Fifth, both detected events targeted and general are extended to get more location information. Finally, the events related tweets with spatio-temporal information are visualized by using a map visualization.

We use Apache Spark platform [1] to do heavy computation with huge data. Since spark is an in-memory computation platform, spark has better speedup to process big data in parallel, compared with other parallel data processing such as Hadoop map reduce. We use spark for data processing and classifier stages. For data pool, where all machine processors take the acquired data for further processing, we utilize the power of Fujitsu Exabyte File System (FEFS). It is a parallel file storage system technology. FEFS is a software for HPC cluster systems, developed by Fujitsu Ltd. It enables high-speed parallel distributed processing of huge amounts of transactions [7]. As well as, it has superior features such as actual operational convenience, system scalability, and high reliability for zero operational downtime during a long computation. Thus, it contributes to significant improvements in system performance. Those FEFS and spark technologies are installed on top of HPC cluster.

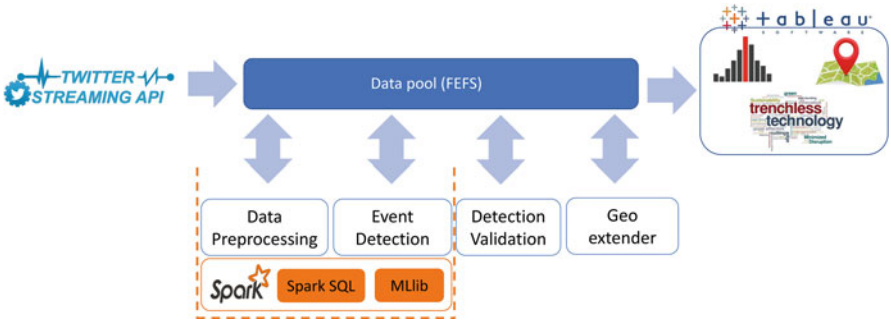


Fig. 3.1 Spatio-temporal events detection: the workflow

3.3.1 Data Acquisition

We use social media data source (twitter) related to traffic. It is done by defining a set of keywords and a set of twitter user accounts which tend to post messages relevant to traffic such as government and media user accounts. Data crawling is performed by invoking Twitter streaming API through a java-based crawler application. The acquired data subsequently be stored in a data pool as raw data in FEFS system, which has been described in Sect. 3.3. Any further data processing will pull raw data from this pool.

Dataset Structure

The acquired data is in raw JavaScript object notation (JSON) as a Twitter data format. It is stored in a file system as JSON file extension. In raw format, each status message contains a bunch of attributes. For our experiment purposes, we use several selected required attributes for spatio-temporal event detection. The structure of raw and extended status message is shown in Tables 3.1 and 3.2, respectively.

The illustration of selected fields of raw twitter JSON data is shown in Fig. 3.2. It is delimited by "|" character for each attribute.

After the data processing, classification, and geo-extender function are applied, it extends additional attributes for spatio-temporal purposes, in order to easily plot status message's location on map visualization, as shown in Table 3.2.

Each attribute is defined as follows:

- Created_at: the time when the status message is posted by user (timestamp).
- Latitude, Longitude: geolocation of status message.
- Text: the message content posted by user.

Table 3.1 Raw status message data structure

Attribute	Length	Data type
Created_at	30	Time stamp
Latitude		Double
Longitude		Double
Text	140	String

Table 3.2 Extended status message data structure

Attribute	Length	Data type
Created_at	30	Time stamp
Latitude		Double
Longitude		Double
Text	140	String
Postal_code	8	String
Type	100	String

Wed Aug 21 16:54:04 +0000 2017 | 51.5131392 | -0.1397757 | Having a lovely day sitting in grid lock traffic...
Wed Aug 22 08:31:09 +0000 2017 | 51.4943053 | -0.1023826 | Traffic at this time, that's London for you...
Wed Aug 23 09:01:54 +0000 2017 | null | null | stuck in London bcose it outage i called u following...
Wed Aug 23 10:15:18 +0000 2017 | null | null | Victoria park hackney hackneytower hamlets london...
Wed Aug 23 10:29:07 +0000 2017 | 51.3898872 | 0.0421551 | Southborough a26 London road slow moving traffic...
Wed Aug 24 15:22:41 +0000 2017 | null | null | on way home from london; been delayed in traffic...
Wed Aug 25 10:29:27 +0000 2017 | 51.5088869 | -0.1140182 | Actually hate london traffic it's taken 30 mins to go...
Wed Aug 26 12:45:32 +0000 2017 | 51.46297071 | -0.5004316 | m25 delays near j14/a3113 anticlockwise caused...
Wed Aug 28 05:09:11 +0000 2017 | null | null | fun fact london underground's logo was inspired by what is on...
Wed Aug 28 17:00:51 +0000 2017 | null | null | did you know the london chatham and dover railway opened...

Fig. 3.2 Illustration of raw twitter JSON data

2017-08-21-16 | 51.5131392 | -0.1397757 | Having a lovely day sitting in grid lock traffic... | W1B | route
2017-08-22-08 | 51.4943053 | -0.1023826 | Traffic at this time, that's London for you... | SE11 | route
2017-08-23-10 | 51.3898872 | 0.0421551 | Southborough, a26 London road slow moving traffic... | BR2 | route
2017-08-25-10 | 51.5088869 | -0.1140182 | Actually hate london traffic it's taken 30 mins to go... | SE1 | route
2017-08-26-12 | 51.4629707 | -0.5004316 | m25 delays near j14/a3113 anticlockwise caused... | TW19 | route

Fig. 3.3 Illustration of processed data after applying data processing, classification, and geo-extender function

- Postal_code: the postal/zip code of status message, e.g., "SE6."
- Type: the location type of detected road name from text attribute, e.g., "route," "point_of_interest."

The illustration of data after applying data processing, classification, and geo-extender function is shown in Fig. 3.3. It is delimited by "|" character for each attribute.

3.3.2 Data Preprocessing

Data preprocessing is the first action against the acquired data in the data pool. Since it has a significant impact on accuracy and quality of learning the data by machine, it is an essential stage in big data analytics workflow [3]. In fact, social media status text contains lots of noise. It has plenty of unnecessary characters and words such as URL, user mention, illegal character, e.g., '&,' punctuation, and stop word. Therefore, the raw data should be preprocessed to clean those up from outliers and make it standard. We utilize spark SQL and regular expression function to preprocess the data, by referring to our defined stop word dictionary, which is adopted from stop word list website [68–70]. Data preprocessing also includes data extraction and parsing such as "created_at" field as the date time posting to get the formatted date time, and "coordinates" field as location precision of status message to get the spatio-temporal information. As well as, tokenization technique is used to

Algorithm: Data Transformation and Cleansing

Input : $twt_i \leftarrow$ raw tweet dataset
Output: cleaned and parsed data of
$twt_i < created\_at, text, latitude, longitude >$

$tw \leftarrow tuple < twt_i.created\_at, twt_i.text, twt_i.coordinates[\varphi, \lambda] >;$
$idx \leftarrow 0;$
foreach $i \leftarrow tw.iterate()$ **do**
  $i \leftarrow i(1).replaceAll("\backslash n","").replaceAll("\&[a-z]+;","");$
  $i \leftarrow i(1).replaceAll("https?://.*.*.\{3\}","").replaceAll("https?://.*[\backslash s\backslash n\backslash r]","");$
  $i \leftarrow i(1).replaceAll("@[a-zA-Z0-9\_]+","");$
  $i \leftarrow i(1).replaceAll("-","").replaceAll("=","").replaceAll("\#","").replaceAll("\.","").replaceAll("|","").replaceAll("!","").replaceAll("","","").replaceAll("<","").replaceAll("","").replaceAll("@","").replaceAll(",","").replaceAll(";","");$
  $tw_{idx} \leftarrow i;$
  $idx \leftarrow idx + 1;$
end
$idx \leftarrow 0;$
foreach $i \leftarrow tw.iterate()$ **do**
  if $i(2) \neq null$ **then**
    $tw \leftarrow tuple < i(0), i(1), i(2)[0], i(2)[1] >;$
  else
    continue();
  end
end
$idx \leftarrow 0;$
foreach $i \leftarrow tw.iterate()$ **do**
  if $i(0) \neq null$ **then**
    $inDateForm \leftarrow pattern("EEE\ MMM\ dd\ HH:mm:ss\ Z\ yyyy");$
    $outDateForm \leftarrow pattern("yyyy-MM-dd-HH");$
    $formated \leftarrow outDateForm.parse(inDateForm);$
    $tw \leftarrow tuple < formated, i(1), i(2), i(3) >;$
    $tw_{idx} \leftarrow i;$
    $idx \leftarrow idx + 1;$
  else
    continue();
  end
end
tw.saveAsFile("path")

Fig. 3.4 Data cleansing and transformation algorithm

transform a set of words or sentences from the tweet into unit pieces called token using predefined separator including whitespace and punctuations. Furthermore, the preprocessed data is used to feed supervised machine learning for classification. Note that we ignore retweet (repost of another user's post) status message, because, it contains the same information. Thus, it leads to efficient processing. The result of data processing is stored back in data pool as cleaned data. Figure 3.4 gives the data cleansing and parsing algorithm. Figure 3.5 shows the algorithm for tokenization and stop word removal.

Algorithm: Tokenization and Stop Words Removal

Input : $twt_i \leftarrow$ cleaned dataset
$twt_i < created\_at, text, latitude, longitude >$,
$stopG \leftarrow$ global stop word dictionary,
$stopT \leftarrow$ context stop word dictionary

Output: tokenized and cleaned data of twt_i from stop words

```
1  stopWords ← stopG_i.union(stopT_i);
2  stopWords ← stopWords.toLowerCase();
3  idx ← 0;
4  token[] ← {};
5  foreach i ← twt_i.iterate() do
6  |  token.append(i(1).splitBy("\W+"));
7  end
8  idx ← 0;
9  remStopW[] ← {};
10 foreach i ← token.iterate() do
11 |  if i.notExist(stopWords) then
12 |  |  remStopW_idx ← i;
13 |  |  idx ← idx + 1;
14 |  else
15 |  |  continue();
16 |  end
17 end
18 remStopW.saveAsFile("path")
```

Fig. 3.5 Algorithm for tokenization and stop word removal

3.3.3 Event Detections

We implemented two types of event detections, they are targeted event and general event. The targeted events are events which are important and become focus of detection, such as traffic in our case. Whereas, general event is any event which occurs at any place and is being discussed in social media corpus. We use supervised learning to detect targeted event, and word frequency analysis to detect general events. Furthermore, the result of general event detections is verified by retrieving information from the Internet, which will be discussed in section **Word Frequency Analysis and Validation**.

Supervised Learning

This supervised learning is used to detect targeted events. In our case, it is traffic event detection which classifies tweet corpus into either traffic-related or non-traffic-related. We built a supervised model to predict the class.

Feature Extraction

Twitter is a text type data which contains sentences. Therefore, we utilize text-feature extraction (also known as bag-of-words representation) to extract features from the corpus. The bag-of-words approach treats a piece of text content as a set of words, and possibly numbers in the text. The process of text-feature extraction in this thesis is tokenization, stop words removal, and vectorization in a row. Tokenization and stop words removal have been discussed in Sect. 3.3.2. While vectorization is the last stage in text-feature extractions, it turns the processed terms into a vector representation. We use term frequency-inverse document frequency (TF-IDF) method for feature extraction. TF-IDF is a widely used method in text mining, which can be used for feature extraction, and perform good result for learning.

We use the hashing technique (HashingTF), which is available in Spark through MLlib. It works by mapping a raw word into an index (term) by applying a hash function. The term frequencies are calculated using the mapped indices. Denote a term by t, a document by d, and the corpus by D. Term frequency, $TF(t,d)$ is the number of times that term t appears in the document d, while document frequency $DF(t,D)$ is the number of documents that contain the term t. If we only use term frequency to measure the importance, it will lead to over-emphasize terms which appear very often but carry little information about the document, e.g., "the," "or," and "a." If a term appears very often in the corpus, it means it does not carry special information about a particular document [71]. Inverse document frequency (IDF) is a numerical measure of how much information a term provides. IDF is calculated by using $IDF(t,D) = \log|D|/DF(t,D)$, where $|D|$ denotes the total number of documents in the corpus. Since logarithm is used, IDF will return 0 if a term appears in all documents. Thus, the TF-IDF measure is calculated by multiplying TF(t,d) and IDF(t,D), that is $TFIDF(t,d,D) = TF(t,d).IDF(t,D)$.

Classification

The goal of classification is to predict the categorical labels of a given new input according to the learning phase in the past. We utilize classification algorithms to detect specific targeted event such as traffic event detection. We applied a sort of binary classification, which categorizes the tweets into two classes, traffic-related or non-traffic-related. In order to get the best model which fits with traffic event detection purposes, we compared the performance of three classification models (logistic regression, Support Vector Machine, and Naïve Bayes) as shown in Table 3.3. The comparison is according to the model evaluation method including evaluation metrics (prediction accuracy, area under precision and recall, and area under ROC). The detail of model evaluation methods and result is explained in section *Accuracy Evaluation.* Furthermore, the model with best performance will be used for classifying the real-world twitter data related to London. Figure 3.6 gives the classification algorithm.

Table 3.3 Performance comparison of three classification models

Model	Prediction accuracy	Area under PR	Area under ROC
SVM	73.395	81.461	74.204
LR	**78.734**	**84.706**	**78.825**
NB	70.721	77.349	71.955

Algorithm: Classification

Input : $twt_i \leftarrow$ preprocessed new data,
$trn \leftarrow$ training data
Output: tweets related to traffic

```
1  tr ← trn.label = 1;
2  ntr ← trn.label = 0;
3  featTraff ← extracting features of tr;
4  featNtraff ← extracting features of ntr;
5  lbTraff ← labeling featTraff = 1;
6  lbNtraff ← labeling featNtraf = 0;
7  trn ← lbTraff.union(lbNtraff);
8  trn.cache();
9  mod ←< ClassificationAlgorithm >.train(trn);
10 idx ← 0;
11 foreach i ← twt_i.iterate() do
12 |   predicted ← mod.predict(i);
13 |   traffic_idx ← predicted.get(1);
14 |   idx ← idx + 1;
15 end
16 traffic.saveAsFile("path")
```

Fig. 3.6 The Classification Algorithm

We use MLlib in Apache Spark to build and train a model using parallel computing. First, we trained the model with more than 1000 training data with its labels for learning purposes. Label 1 denotes traffic-related, and label 0 for non-traffic-related. Secondly, we built and trained three models with default input parameters. The model learns from the training data, and finds the pattern from labels of each tweet text in the training data. Thirdly, we evaluate the model's accuracy by utilizing cross-validation approach with evaluation metrics using testing data. Fourthly, using the best selected model, we predict the labels of new tweets and categorize them into two categories (0 and 1) iteratively. Finally, we filter out the tweets which are not related to traffic for further processing. Furthermore, we summarize the data for analysis purposes and to get the insight by applying several data summaries such as counting number of tweets with hourly basis, and plotting for displaying location dissemination of traffic events.

Accuracy Evaluation

We need to know how well our model performs, especially when dealing with unseen data. First, we use cross-validation approach to divide the dataset into training and testing data. Finally, we evaluate the prediction results of testing data by utilizing evaluation metrics. We compared the performance of three classification models (logistic regression, support vector machine, Naïve Bayes). Furthermore, we select the best model to predict the real-world twitter data. We use train–test split with ratio 80/20 for training set and testing set. It is a good starting point for splitting technique according to the literature.

Cross-Validation

We use a train–test split, which is one of the cross-validation evaluation approaches. It is straightforward, yet effective for validation purposes. We divide our dataset into two non-overlapping parts (training set and testing set). Training set is used to train our model, whereas testing set or hold-out set is used to evaluate the performance of our models when dealing with unseen data using evaluation measurement. We use various training/testing split ratios of the dataset in our experiment. These splitting ratios include 50/50, 60/40, 70/30, and 80/20 ratios.

Evaluation Metrics

The performance of models when predicting the class/label of testing set (unseen data) is measured by utilizing evaluation metrics, which is commonly used in binary classification. The evaluation metrics include prediction accuracy, the area under the precision-recall curve, and the area under ROC curve (AUC). It is according to cross-validation using train-test split which was discussed in section *Cross-validation*.

Word Frequency Analysis and Validation

We built a workflow as shown in Fig. 3.7. in order to detect general event detections, as well as the validation of detected events. The used technique is word frequency analysis. The workflow begins by pulling the cleaned and tokenized data, generating the most frequent words among the whole tweets yields in a list of event candidates, checking the validity of each event candidate by scrapping a London government's website (london.gov.uk), and finally resulting in a list of occurred events with its detail, time, and location. The London government website/source is used for proof of concept purposes. Future work will look into building up a list of resources and methods for validation. Figure 3.8 gives the master algorithm for the validation of the detected events.

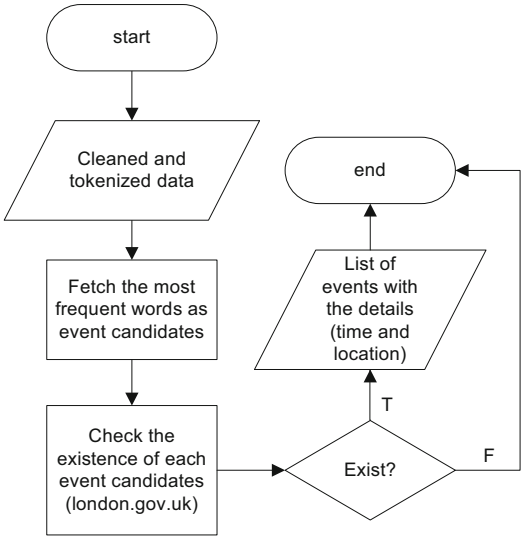


Fig. 3.7 Workflow of general event detection

The data preprocessing process includes data cleansing (meaningless object removal, stop word removal) and tokenization. The most frequent words (*fw*) are generated by counting each word occurrence (*wo*) in the tweets collection, then, sorting them by *wo* in descending, and picking ten terms in the first list. The first ten terms are chosen according to the defined threshold *n*, which represents the number of *wo*. If the *wo* is greater or equal to *n,* then it is listed in the first ten terms. The size of *n* is adjusted until the size of *fw* is equal to ten. The list of chosen terms in *fw* is considered as a list of event candidates (*ev*).

Furthermore, in order to validate *ev,* we created two functions, *googleSearch* (*gs*) and *webExtractor* (*wx*). The algorithms for these functions are given in Figs. 3.9 and 3.10, respectively. We pass a parameter with pattern "<*fw*> London <current_year>" to *gs,* and *gs* will invoke google search API to find the pattern as a keyword. Thereafter, *gs* returns a list of URLs related to the pattern, if the list contains london.gov.uk, it means the event exists and has been detected. Finally, function *wx* extracts the events information including event name, time, location, ticketing, and description from london.gov.uk. Therefore, we only detect major events in London which are listed in london.gov.uk. The functions *gs* and *wx* are python-based program, which is bundled as one function *searchScrape* (*ss*), and requires four input parameters, which are file path of list of *fw,* the trusted link to get event information, minimum size of *fw,* and threshold number of *n.* The output of *ss* function is a list of detected events and its information such as event name, time (start-end date), event location, tickets, and event description.

Algorithm: Validation of Detected Events

Input : $fw_i \leftarrow$ list of most frequent words,
$trustLink \leftarrow$ trusted link for the ground truth,
$minTerm \leftarrow$ number of desired event candidates,
$thr \leftarrow$ threshold to satisfy $minTerm$

Output: $ev_k \leftarrow$a list of detected events with its detail

```
1  ev_c = event candidates;
2  foreach term ← fw_i.iterate() do
3  |  tm ← term.split(0);
4  |  n ← term.split(1);
5  |  if n >= thr then
6  |  |  ev_c.append(tm)
7  |  else
8  |  |  continue();
9  |  end
10 end
11 // check that # event candidates meets the request;
12 if ev_c.length() < minTerm then
13 |  print "please reduce the threshold to get minTerm of event
   |   candidates";
14 |  sys.exit()
15 else
16 |  continue();
17 end
18 // search and scrape;
19 eventInfo_i ← a list of events with details;
20 foreach t ← ev_c.iterate() do
21 |  patt ← "t london currentYear()";
22 |  gooResult_j ← googleSearch(patt);
23 |  foreach link ← gooResult_j.iterate() do
24 |  |  if trustLink in link then
25 |  |  |  info ← webExtractor(link);
26 |  |  |  if !info.isEmpty() then
27 |  |  |  |  eventInfo_i ←.append(info);
28 |  |  |  |  eventInfo_i ←"=====";
29 |  |  |  else
30 |  |  |  |  continue();
31 |  |  |  end
32 |  |  else
33 |  |  |  continue();
34 |  |  end
35 |  end
36 end
37 eventInfo_i.saveAsFile();
```

Fig. 3.8 Validation of the Detected Events: The Master Algorithm

```
Algorithm: googleSearch
   Input  : inPatt ← an input pattern as keyword search
   Output: url_i ← a list of URLs related to the keyword
 1 resp ←request.get(url.encode('utf-8'));
 2 objWeb ← resp.parseHTML();
 3 url_i ← [ ];
 4 foreach obj ← objWeb.findAllURLs() do
 5 |   found ←
   |     regex.search("(http|ftp|https ):////([\w_-]+(?:(?:.[\w_-]+)+))([\w ., @?^=%&:/~+#-]*[
   |     \w@?^=%&/~+#-])?&sa");
 6 |   if found.exist() == true then
 7 |   |   url_i.append(found.group(0).replace('&sa',''))
 8 |   else
 9 |   |   continue();
10 |   end
11 end
12 return url_i
```

Fig. 3.9 Google Search Algorithm

```
Algorithm: webExtractor
   Input  : inURL ← a URL site
   Output: eventInfo ← detail information about detected events
 1 resp ←request.get(inURL.encode('utf-8'));
 2 objWeb ← resp.parseHTML();
 3 cont ← objWeb.find('div', attrs='class':
     'node–event–full–body–wrapper');
 4 ev ← cont.find('h1').text;
 5 startDate ← cont = content.find('time', attrs='class':
     'start-date-time');
 6 endDat ← cont.find('time', attrs='class': 'end-date-time');
 7 address ← cont.find('span', attrs='class': 'address-container');
 8 tickets ← cont.find('span', attrs='class': 'tickets');
 9 desc ← cont.find('div', attrs='class': 'field
     field–name-field-event-stub-info field–type-text-long
     field–label-hidden');
10 eventInfo.append(ev, startDate, endDate, address, tickets, desc);
11 return eventInfo
```

Fig. 3.10 The Web Extractor Algorithm

3.3.4 Geo-Extender

In order to find the geographical distribution of traffic status using tweets in the form of a geographical map, we need to get geographic location of traffic-related status messages in the form of Cartesian coordinates (latitude, longitude). This will help in analyzing the tweets and extracting useful information. This process

is done by invoking Google Maps Geocoding API. It is a web service provided by Google Inc. which provides geocoding and reverse geocoding of given addresses [8]. Geocoding is a process of converting given addresses (i.e., a street address) into geographic coordinates (latitude, longitude), which can be used to pinpoint a location of given input on a map, or position on the map. The reverse geocoding facilitates the opposite. It converts given geographic coordinates into a human-readable address. Reverse geocoding will provide the detail of location information of the given point, which is easy to read and understand by humans, such as the postal code, road name, city, street number, and district.

We take the tweet data and the geo location information including the postal code as a data source of Tableau. Moreover, we generate several visualization summaries such as a graph, word cloud, and map. With a graph, we can see the number of tweets with the time-frequency distribution in order to see the anomaly which indicates higher traffic than the usual. With word cloud, we can see the most mentioned words which imply the top hot topics. With a map, we can depict the dissemination of traffic condition on a particular area. We plot the location spreading and its intensity of tweets related to the road traffic in London.

3.3.5 Analysis and Visualization

There are many ways to plot geolocation data into a map visualization for analysis purposes. One of them is by using Tableau software. Tableau is a business intelligence software which helps people to see and have a better understanding of their data [2]. It enables users to explore their data with limitless visual analytics. As well as, it eases user to perform ad-hoc analysis with just a few clicks. We take the processed and geolocated data as a data source of Tableau; then we generate several visualization summaries such as a graph, word cloud, and map. With a graph, we can see the number of tweets with the time-frequency distribution in order to see the anomaly which indicates, for instance, higher traffic than usual. With word cloud, we can see the most mentioned words which imply the hot topic. With a map, we can depict the dissemination of traffic conditions on a particular area.

3.4 Result and Discussion

For experiment purposes, we gathered twitter data between 21st August and 13th September 2017, represented as hourly data in the range (0, 576). In total, it consists of three million records of tweet related to our defined keyword. However, after classification process which aims to filter out non-related traffic tweet, the number of tweets was reduced to a smaller number. This reduction in the tweet count is expected, as these status messages are not genuinely related to traffic. Even though it contains a traffic-related word, it does not mean a road traffic problem. Figure 3.11

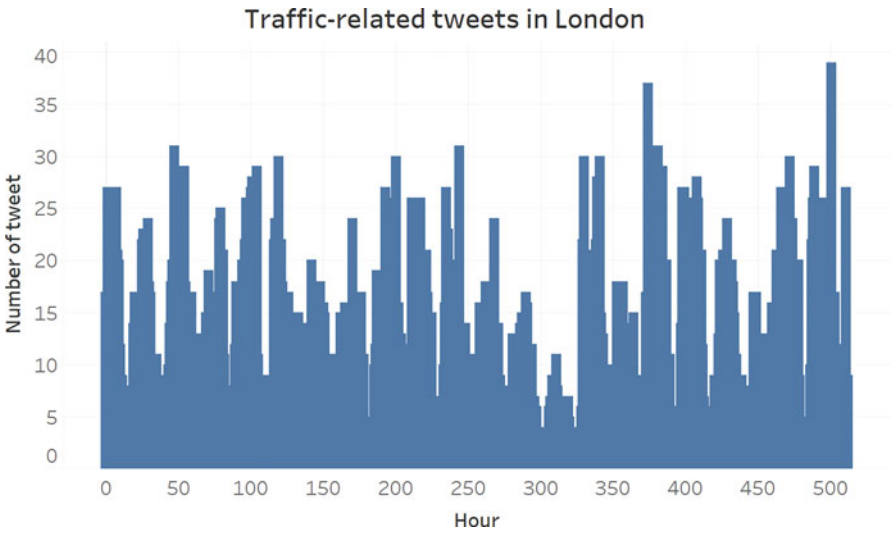


Fig. 3.11 Hourly number of tweets related to traffic in London

shows that the time-frequency distribution of the number of tweets is varied hourly. The peak hours are around 375th and 500th hours. By looking at this, we get an indication that there might be an ongoing event occurring at that time which affected the traffic condition in London.

The location-intensity distribution of the number of tweets related to traffic in London is shown in Fig. 3.12. Red color level varies the number of distribution. The higher the intensity of red, the more traffic an area has. The figure shows that the areas around downtown had more traffic. We assume that an increase in the number of traffic-related tweets indicates higher intensity of road traffic. Grey color denotes the areas where we could not get geotagged tweets since not all acquired tweets are geotagged. There are three main areas with high-intensity traffic (the red color) on the map. These are in South Bank (shown as 1), around Greenwich park (shown as 2), around Crystal Palace Park and National Trust-Morden Hall Park (shown as 3), in Fig. 3.12. The postal codes are SE1, SE10, SE19, and SM4, respectively.

According to the London events calendar [72], in south bank, there has been an event held, called "underbelly festival." It has started from 28th April to 30th September 2017. It was a festival event held by Underbelly, which showcases cabaret, comedy, live circus, and family entertainment [9]. In another hand, around Greenwich Park, Crystal Palace Park, and National Trust-Morden Hall Park, there have been another events, called "The Luna Cinema" [10] running from June to October 2017. It is an outdoor cinema for citizens or tourist to spend their warm summer evening watching a new film release or age-old classics.

Moreover, to detect the occurred events automatically, we built a workflow as described in section **Word Frequency Analysis and Validation**. By using the ss function, we retrieved a list of detected events and their information such as event

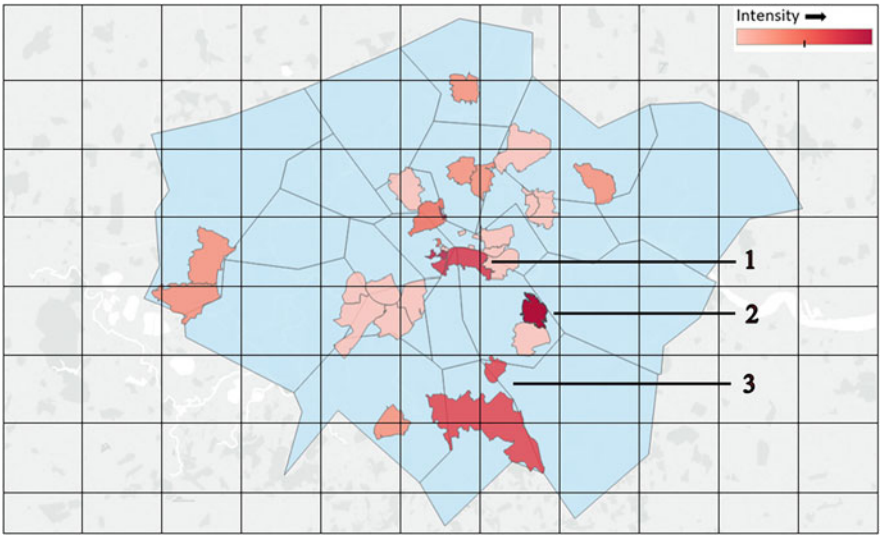


Fig. 3.12 Tweet intensity related to traffic in London

eventinformation.txt

```
Event name| Notting Hill Carnival 2017
Start date| Saturday 26 August 2017
End date| Monday 28 August 2017
Address| Notting Hill Notting Hill, London, W11, United Kingdom
Tickets| No tickets required
Event Description|
The Notting Hill Carnival, Europe's biggest street festival is organised by London Notting Hill Carnival Enterprises
```

Fig. 3.13 Details of a detected event

name, time (start-date and end-date), event location, tickets, and event description. This is shown in Fig. 3.13. for the Notting Hill Carnival event.

In conclusion, we detected an event by a term "carnival," which is among the most frequent words. The event name is called "Notting Hill Carnival 2017." The location spreading of tweets related to the carnival are shown in Fig. 3.14. Most of the tweets come from one area, as shown by the red color intensity in the figure. By inspecting this circumstance, we can infer that there was an event related to the carnival on that area. This red color area lies around Notting Hill London, and was the location of Notting Hill carnival [12], the Europe's biggest street festival which is organized by London Notting Hill carnival enterprises trust.

The daily frequency distribution of the number of tweets about the carnival is shown in Fig. 3.15. We can see that the peak is on the 28th August, while it is increasing gradually from 26th August and decreasing by the 30th August. By observing this phenomenon, we can conclude that around those dates, there was an event related to a carnival. The carnival event was held on 26th–28th August 2017 in London [12].

This information about the event was detected automatically without any prior knowledge of the event, its location and time.

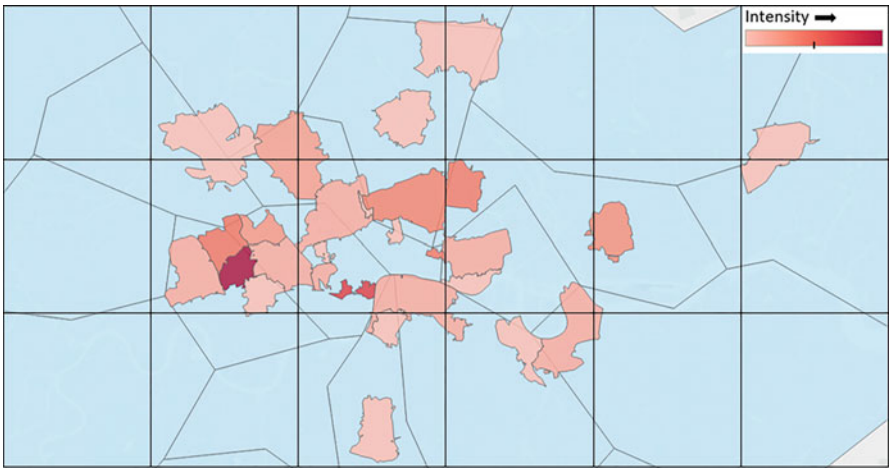


Fig. 3.14 Tweet intensity related to Notting Hill carnival in London

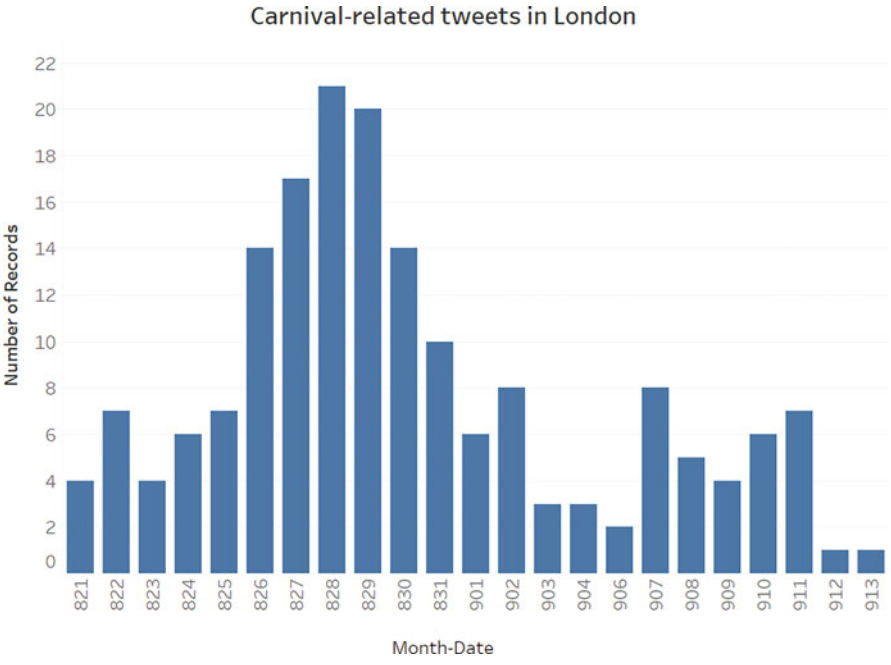


Fig. 3.15 Number of tweets in the period related to Notting Hill carnival in London

3.5 Conclusion

Social media have revolutionized our societies and are gradually becoming a key pulse of smart societies by sensing the information about the people and their spatio-temporal experiences around the living spaces. Analyzing social media data such as twitter has become a cost-effective way to detect events, by utilizing big data technologies such as spark, FEFS, and tableau. In this paper, we use Twitter for the detection of spatio-temporal events in London. Specifically, we use big data and AI platforms including Spark, and Tableau, to study twitter data about London.

We have empirically demonstrated that events can be detected automatically by analyzing Twitter data. We detect the occurrence of multiple events including "Underbelly festival" and "The Luna Cinema." The Underbelly festival was located at South Bank, while The Luna Cinema was located in multiple places including around Greenwich Park, Crystal Palace Park, and National Trust-Morden Hall Park. We have also detected the London Notting Hill Carnival 2017 event, the Europe's biggest street festival which was organized by London Notting Hill carnival enterprises trust. This was located around Notting Hill [12]. We have detected the locations and times of these events automatically, without any prior knowledge of the events. The results presented in the paper have been obtained by analyzing over three million tweets. The textual analysis of the tweets was used to ensure that the tweets are truly related to the road traffic. We have also used the integration of big data technologies with HPC to enhance scalability and computational intelligence.

This paper has made the following specific enhancements over our earlier work [3, 4]. It has provided a comparison of three machine learning methods, support vector machine, logistic regression, and Naïve Bayes for event detection purposes using various performance metrics. It has introduced an enhanced methodology to automatically validate the factuality of the detected events. The validation methodology is used to confirm that the events, which were detected by our system, did actually happen at the detected time and place. This paper elaborates on the methodology and architecture of the event detection and validation system and provides algorithms for the main components of the proposed system. Moreover, an extended literature review has been added to this paper.

We have improved the data management and processing methodology compared to the earlier versions. However, it still needs improvements to have better detection accuracy, wider spatio-temporal detection, and better quality of analysis. For better detection accuracy, we plan to continue to enhance our automatic validation methodology and compare the result with actual information by associating it with events reporting such as news or media websites. For wider spatio-temporal event detection, we would consider additional social media data such as Facebook. For better quality of analysis, we hope to utilize better AI techniques.

Acknowledgments The authors acknowledge with thanks the technical and financial support from the Deanship of Scientific Research (DSR) at the King Abdulaziz University (KAU), Jeddah, Saudi Arabia, under the grant number G-651-611-38. The experiments reported in this paper were performed on the Aziz supercomputer at KAU.

References

1. Apache Software Foundation: Apache Spark, https://spark.apache.org/
2. Tableau: What Is Tableau - Make Your Data Make an Impact, https://www.tableau.com/trial/tableau-software
3. Suma, S., Mehmood, R., Albugami, N., Katib, I., Albeshri, A.: Enabling next generation logistics and planning for smarter societies. Procedia Comput. Sci. **109**, 1122–1127 (2017)
4. Suma, S., Mehmood, R., Albeshri, A.: Automatic event detection in smart cities using big data analytics. In: International Conference on Smart Cities, Infrastructure, Technologies and Applications SCITA 2017: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 111–122. Springer, Cham (2018)
5. Usman, S., Mehmood, R., Katib, I.: Big data and HPC convergence: the cutting edge and outlook. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 11–26. Springer, Cham (2018)
6. Aziz Supercomputer. Top500, https://www.top500.org/site/50585
7. Fujitsu Ltd.: Fujitsu Releases World's Highest-Performance File System, http://www.fujitsu.com/global/about/resources/news/press-releases/2011/1017-01.html
8. Google Inc.: Getting Started | Google Maps Geocoding API | Google Developers, https://developers.google.com/maps/documentation/geocoding/start
9. Underbellyfestival.com: About underbelly festival, http://www.underbellyfestival.com/about
10. The Luna Winter Cinema 2018, https://thelunacinema.com/
11. Wikipedia: Notting Hill Carnival, https://en.wikipedia.org/wiki/Notting_Hill_Carnival
12. London.gov.uk: Notting Hill Carnival 2017, https://www.london.gov.uk/events/2017-08-26/notting-hill-carnival-2017
13. Mehmood, R., Alam, F., Albogami, N.N., Katib, I., Albeshri, A., Altowaijri, S.: UTiLearn: a personalised ubiquitous teaching and learning system for smart societies. IEEE Access. **3536**, 1–22 (2017)
14. Tawalbeh, L., Basalamah, A., Mehmood, R., Tawalbeh, H.: Greener and smarter phones for future cities: characterizing the impact of GPS signal strength on power consumption. IEEE Access. pp, 1–1 (2016)
15. Graham, G., Mehmood, R.: The strategic prototype "crime-sourcing" and the science/science fiction behind it. Technol. Forecast. Soc. Change. **84**, 86–92 (2014)
16. Tawalbeh, L.A., Bakhader, W., Mehmood, R., Song, H.: Cloudlet-based mobile cloud computing for healthcare applications. In: 2016 IEEE Global Communications Conference (GLOBECOM). pp. 1–6. IEEE (2016)
17. Muhammed, T., Mehmood, R., Albeshri, A., Katib, I.: UbeHealth: a personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities, (2018)
18. Alotaibi, S., Mehmood, R.: Big data enabled healthcare supply chain management: opportunities and challenges. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 207–215. Springer, Cham (2018)
19. Alamoudi, E., Mehmood, R., Albeshri, A., Gojobori, T.: DNA profiling methods and tools: a review. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 216–231 (2018)

20. Khanum, A., Alvi, A., Mehmood, R.: Towards a semantically enriched computational intelligence (SECI) framework for smart farming. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 247–257. Springer, Cham (2018)
21. Aqib, M., Mehmood, R., Albeshri, A., Alzahrani, A.: Disaster management in smart cities by forecasting traffic plan using deep learning and GPUs. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 139–154 (2018)
22. Alam, F., Mehmood, R., Katib, I.: D2TFRS: An object recognition method for autonomous vehicles based on RGB and spatial values of pixels. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST. pp. 155–168. Springer, Cham (2018)
23. Muhammed, T., Mehmood, R., Albeshri, A.: Enabling reliable and resilient IoT based smart city applications. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 169–184. Springer, Cham (2018)
24. Mehmood, R., Faisal, M.A., Altowaijri, S.: Future networked healthcare systems: a review and case study. In: Boucadair, M., Jacquenet, C. (eds.) Handbook of Research on Redesigning the Future of Internet Architectures, pp. 531–558. IGI Global, Hershey, PA (2015)
25. Mehmood, R., Graham, G.: Big data logistics: a health-care transport capacity sharing model. Procedia Comput. Sci. **64**, 1107–1114 (2015)
26. Arfat, Y., Mehmood, R., Albeshri, A.: Parallel shortest path graph computations of United States road network data on apache spark. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 323–336. Springer, Cham (2018)
27. Mehmood, R., Meriton, R., Graham, G., Hennelly, P., Kumar, M.: Exploring the influence of Big Data on city transport operations: a Markovian approach. Int. J. Oper. Prod. Manag. Forthcomin, (2016)
28. Mehmood, R., Crowcroft, J., Hand, S., Smith, S.: Grid-level computing needs pervasive debugging. In: Proceedings - IEEE/ACM International Workshop on Grid Computing. pp. 186–193 (2005)
29. Adamson, M.K., Mehmood, R.: Developing event based hierarchical middleware for E-learning. In: Miguel Baptista Nunes and Maggie McPherson (series editors: Piet Kommers, P.I. and N.-S.C. (ed.) Proceedings of the IADIS International Conference on e-Learning (e-Learning 2007). pp. 284–291. IADIS, Lisbon, Portugal (2007)
30. Büscher, M., Coulton, P., Efstratiou, C., Gellersen, H., Hemment, D., Mehmood, R., Sangiorgi, D.: Intelligent mobility systems: Some socio-technical challenges and opportunities. In: Communications Infrastructure. Systems and Applications in Europe, Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST 16. pp. 140–152 (2009)
31. Schlingensiepen, J., Nemtanu, F., Mehmood, R., McCluskey, L.: Autonomic Transport Management Systems—Enabler for Smart Cities, Personalized Medicine, Participation and Industry Grid/Industry 4.0. In: Intelligent Transportation Systems – Problems and Perspectives, Volume 32 of the series Studies in Systems, Decision and Control. pp. 3–35. Springer International Publishing (2016)
32. Schlingensiepen, J., Mehmood, R., Nemtanu, F.C.: Framework for an autonomic transport system in smart cities. Cybern. Inf. Technol. **15**, 50–62 (2015)
33. Schlingensiepen, J., Mehmood, R., Nemtanu, F.C., Niculescu, M.: Increasing sustainability of road transport in European cities and metropolitan areas by facilitating autonomic road transport systems (ARTS). In: Wellnitz, J., Subic, A., Trufin, R. (eds.) Sustainable Automotive Technologies 2013 Proceedings of the 5th International Conference ICSAT 2013, pp. 201–210. Springer International Publishing, Ingolstadt, Germany (2014)

34. Mehmood, R., Nekovee, M.: Vehicular AD HOC and grid networks: Discussion, design and evaluation. In: 14th World Congress on Intelligent Transport Systems, ITS 2007. pp. 1555–1562 (2007)
35. Gillani, S., Shahzad, F., Qayyum, A., Mehmood, R.: A survey on security in vehicular ad hoc networks. (2013)
36. Alvi, A., Greaves, D., Mehmood, R.: Intra-vehicular verification and control: a two-pronged approach. In: 7th IEEE International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2010. pp. 401–405 (2010)
37. Nabi, Z., Alvi, A., Mehmood, R.: Towards standardization of in-car sensors. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), LNCS Volume 6596,. pp. 216–223 (2011)
38. Alazawi, Z., Alani, O., Abdljabar, M.B., Altowaijri, S., Mehmood, R.: A smart disaster management system for future cities. In: Proceedings of the 2014 ACM international workshop on Wireless and mobile technologies for smart cities - WiMobCity '14. pp. 1–10. ACM Press, New York, New York, USA (2014)
39. Alazawi, Z., Abdljabar, M.B., Altowaijri, S., Vegni, A.M., Mehmood, R.: ICDMS: an intelligent cloud based disaster management system for vehicular networks. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), LNCS, Volume 7266. pp. 40–56. Springer, Vilnius, Lithuania (2012)
40. Alazawi, Z., Alani, O., Abdljabar, M.B., Mehmood, R.: An intelligent disaster management system based evacuation strategies. In: 2014 9th International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2014. pp. 673–678 (2014)
41. Ayres, G., Mehmood, R.: On discovering road traffic information using virtual reality simulations. In: 11th International Conference on Computer Modelling and Simulation, UKSim 2009. pp. 411–416 (2009)
42. Mehmood, R.: Towards understanding intercity traffic interdependencies. In: 14th World Congress on Intelligent Transport Systems, ITS 2007. pp. 1793–1799 (2007)
43. Mehmood, R., Lu, J.A.: Computational Markovian analysis of large systems. J. Manuf. Technol. Manag. **22**, 804–817 (2011)
44. Graham, G., Mehmood, R., Coles, E.: Exploring future cityscapes through urban logistics prototyping: a technical viewpoint. Supply Chain Manag. **20**, 341–352 (2015)
45. Ayres, G., Mehmood, R.: LocPriS: a security and privacy preserving location based services development framework. (2010)
46. Naimur Rahman, M., Esmailpour, A., Zhao, J.: Machine learning with big data an efficient electricity generation forecasting system. Big Data Res. **5**, 9–15 (2015)
47. Khan, Z., Anjum, A., Soomro, K., Tahir, M.A.: Towards cloud based big data analytics for smart future cities. J. Cloud Comput. **4**, 1–11 (2015)
48. Luis Felipe Herrera-Quintero, Klaus Banse, Julian vega-Alfonso, A.V.-S.: Smart ITS Sensor for the transportation Planning using the IoT and Bigdata approaches to produce ITS cloud services. 3–9
49. Kolchyna, O., Treleaven, P.C., Aste, T.: A framework for Twitter events detection, differentiation and its application for retail brands. (2016)
50. Arfat, Y., Aqib, M., Mehmood, R., Albeshri, A., Katib, I., Albogami, N., Alzahrani, A.: Enabling smarter societies through Mobile big data fogs and clouds. Procedia Comput. Sci. **109**, 1128–1133 (2017)
51. García, S., Ramírez-Gallego, S., Luengo, J., Benítez, J.M., Herrera, F.: Big data preprocessing: methods and prospects. Big Data Anal. **1**, 9 (2016)
52. Fang, X., Zhan, J.: Sentiment analysis using product review data. J. Big Data. **2**, 5 (2015)
53. Hu, W., Wang, H., Peng, C., Liang, H., Du, B.: An event detection method for social networks based on link prediction. Inf. Syst. **71**, 16–26 (2017)
54. Ma, Z., Yang, Y., Sebe, N., Zheng, K., Hauptmann, A.G.: Multimedia event detection using a classifier-specific intermediate representation. IEEE Trans. Multimed. **15**, 1628–1637 (2013)
55. Rao, A.S., Gubbi, J., Marusic, S., Palaniswami, M.: Crowd event detection on optical flow manifolds. IEEE Trans. Cybern. **46**, 1524–1537 (2015)

56. Doulamis, N.D., Doulamis, A.D., Kokkinos, P., Varvarigos, E.M.: Event detection in twitter microblogging. IEEE Trans. Cybern. **46**, 2810–2824 (2016)
57. Gu, Y., Sean, Z., Chen, F.: From twitter to detector : real-time traffic incident detection using social media data. Transp. Res. Part C Emerg. Technol. **67**, 321–342 (2016)
58. Nguyen, D.T., Jung, J.E.: Real-time event detection for online behavioral analysis of big social data. Futur. Gener. Comput. Syst. **66**, 137–145 (2017)
59. Unankard, S., Li, X., Sharaf, M.A.: Emerging event detection in social networks with location sensitivity. World Wide Web. **18**, 1393–1417 (2015)
60. Wang, Y., Kankanhalli, M.S.: Tweeting cameras for event detection categories and subject descriptors. Int. World Wide Web Conf. Comm. 1231–1241 (2015)
61. Kaleel, S.B., Abhari, A.: Cluster-discovery of twitter messages for event detection and trending. J. Comput. Sci. **6**, 47–57 (2015)
62. Tonon, A., Cudré-Mauroux, P., Blarer, A., Lenders, V., Motik, B.: ArmaTweet: detecting events by semantic tweet analysis. In: Proc. of the 14th extended semantic web conference (ESWC). pp. 138–153. Springer, Champions (2017)
63. Pandhare, K.R., Shah, M.A.: Real Time Road Traffic Event Detection Using Twitter and Spark. Int. Conf. Inven. Commun. Comput. Technol. **2017**, 445–449 (2017)
64. D'andrea, E., Ducange, P., Lazzerini, B., Marcelloni, F.: Real-time detection of traffic from twitter stream analysis. IEEE Trans. Intell. Transp. Syst. **16**, (2015)
65. Hou Lei, K., Khadiwala, R., Chen-Chuan Chang, K.: TEDAS: a Twitter based event detection and analysis system. (2012)
66. Gutierrez, C., Figuerias, P., Oliveira, P., Costa, R., Jardim-Goncalves, R.: Twitter mining for traffic events detection. In: 2015 Science and Information Conference (SAI). pp. 371–378. IEEE (2015)
67. Alomari, E., Mehmood, R.: Analysis of tweets in Arabic language for detection of road traffic conditions. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST. pp. 98–110. Springer, Cham (2018)
68. Ranks.nl: stopwords, http://www.ranks.nl/stopwords
69. Lextek.com: Stop Word List 1, http://www.lextek.com/manuals/onix/stopwords1.html
70. Github.com/Alir3z4: stop-words, https://github.com/Alir3z4/stop-words/blob/master/english.txt
71. Spark, A.: Apache Spark - Feature Extraction and Transformation - RDD-based API, https://spark.apache.org/docs/2.2.0/mllib-feature-extraction.html#tf-idf
72. Visitlondon.com: London Events Calendar, http://www.visitlondon.com/things-to-do/whats-on/special-events/london-events-calendar#KRbiWhui4SMAd9PT.97

Chapter 4
In-Memory Deep Learning Computations on GPUs for Prediction of Road Traffic Incidents Using Big Data Fusion

Muhammad Aqib, Rashid Mehmood, Ahmed Alzahrani, and Iyad Katib

4.1 Introduction

Road transportation is the backbone of modern economies. Unfortunately, every year 1.25 million people die due to road traffic crashes around the globe, equaling a shocking 3400 deaths/day, or 2 deaths/min [1]. Another 20–50 million people suffer injuries annually due to road traffic collisions and many of these incur disability as a consequence of their injuries [1]. These incidents cause great socio-economic and environmental damages globally [2]. Road traffic collisions are the leading cause of deaths of young people aged 15–29 years. Half of the road traffic deaths are of people aged 15–44 years. Certainly, the human loss is the major element here which negatively affects individuals and families, their mental states, motivations, energies, and well-being. Children in broken families are susceptible to becoming criminals or subjects of crimes. The death of family members is likely to affect financial circumstances of the families, depriving children good education and upbringing. These deaths also make our nations lose skills and increase the burden on the social security system.

Road collisions are also a major cause of sudden and unexpected congestion on the road networks. INRIX Research has conducted the biggest study on congestion costs based on the data acquired by 300 million vehicles and devices from 1360

M. Aqib · A. Alzahrani · I. Katib
Department of Computer Science, FCIT, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: mpervez@stu.kau.edu.sa; asalzahrani@kau.edu.sa; iakatib@kau.edu.sa

R. Mehmood (✉)
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: RMehmood@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_4

cities in 38 countries during 2017 [3]. The study revealed that Los Angeles was the worst congested city globally where drivers spent 102 peak hours in congestion, equaling an average of 12% of their total drive time, and the congestion in the city costing $19.2 billion to the drivers and the US economy. The cost of congestion in New York was the highest at $33.7 billion for any single city in the world. The total congestion cost across the US, UK, and Germany was about $461 billion. The cost of congestion to the US economy, alone, exceeded $305 billion. According to a Texas Transportation report [4], 2.9 billion gallons of fuel was wasted in the USA alone during 2012 due to traffic congestion. Traffic congestion also causes air pollution that damages public health and the planet environment [5].

Traffic congestion could be recurrent or nonrecurrent [6]. It could be caused by a demand exceeding the road capacity (e.g., during peak hours) or it is due to different types of incidents or events on the roads such as roadworks. Accidents and crashes are considered a key source of nonrecurrent congestion in up to 60% of the congestion cases on the road networks [7] and these types of congestion are difficult to manage due to their abrupt nature.

Reducing the number of road collisions and better managing their aftermath are of paramount importance to avoid or minimize deaths, injuries, congestion, and other socio-economic losses and environmental damages. One approach to avoid and minimize losses is to automatically predict road traffic incidents and crashes either before they happen or as soon as possible afterward. Traffic management authorities use the so-called incident management systems to manage the situations during and after the incidents. Traffic incident management (TIM) is one such planning process that defines a clearance time between the occurrence of the incident and the removal of all the vehicles, etc., from the incident scene [8].

Many researchers have attempted to address the road incident prediction and management problem. Simulations and modeling have been widely used in the past for event detection on the roads or to foresee the impact of certain events, see, e.g., [9–13]. Several works have emerged in the recent years that use artificial intelligence (AI) and data mining techniques to analyze real road traffic data and predict future traffic characteristics, such as flow, speed, and occupancy [14–20]. Various AI techniques have also been applied to road traffic data for incident prediction [21–23]. Moreover, incident data has been used to predict the duration, spatiotemporal impact, and cost of the incidents by analyzing the incident data [24–34].

Smart infrastructure developments have accelerated the pace of technological advancements and the penetration of these technologies to all spheres of everyday life including transportation [35–39]. The use of GPS devices and mobile signals to collect vehicle location and congestion data, the use of big data [40–42] and high performance computing (HPC) [40, 42–44] technologies, cloud and fog computing [45–49], image processing and artificial intelligence (AI) for traffic analysis, urban logistics prototyping [50], vehicular ad hoc networks [46, 51–54], autonomous driving [55], autonomic transportation systems [56–58], and the use of social media for traffic event detection [59–61] are but a few examples.

This paper brings together transport big data, deep learning, in-memory computing, and GPU computing to predict traffic incidents on the road, thereby providing a novel and comprehensive approach toward large-scale, faster, and real-time incident prediction. Big data refers to the "emerging technologies that are designed to extract value from data having four Vs characteristics; volume, variety, velocity and veracity" [62]. GPUs provide massively parallel computing power to speed up computations. Big data leverages distributed and high performance computing (HPC) technologies, such as GPUs, to manage and analyze data. Big data and HPC technologies are converging to address their individual limitations and exploit their synergies [63]. In-memory computing allows faster analysis of data by the use of random access memories (RAMs) as opposed to the secondary memories. Deep learning is a branch of machine learning that uses hierarchical architectures to learn high-level abstractions in the data [64]. Different deep learning approaches are used to train the models for different purposes including convolutional neural networks (CNNs), recurrent neural networks (RNNs), restricted Boltzmann machines (RBMs), auto-encoders, and more.

We have fused together three different kinds of datasets to predict road traffic incidents. The road traffic dataset provides 5-min interval traffic data on the freeways. It includes vehicle flow, speed, occupancy, the ID of the vehicle detector station (VDS), and other data. The VDS dataset provides data about the vehicle detector stations on the freeways including the identification number of each VDS, location, length, and other attributes. The incident dataset provides information about the recorded incidents. It includes ID of the incident, its location, type, timestamp, duration, and other data about the incidents. The data is acquired from the California Department of Transportation (Caltrans) Performance Measurement System (PeMS) [65].

The fused dataset is used for the training of deep convolution neural networks. Different combinations of the dataset along with different network configurations of the deep learning model are used for training and prediction. The data fusion methodology is explained in detail along with the algorithms. We have analyzed over 10 years of PeMS road traffic data. This work-in-progress paper reports incident prediction results using 3 months' data, September to November 2017. Conclusions are drawn from the current status of the results and ideas for future improvements are given. This paper does not give details of the system architecture, and big data, in-memory, and GPU computing aspects of our software due to the space limitations. These details along with the system architecture can be found in our earlier paper [44] and another chapter to appear in this book [66].

The rest of the paper is organized as follows: Sect. 4.2 discusses the work related to this paper. A detailed discussion on the research methodology including the input dataset and deep model is given in Sect. 4.3. Performance evaluation and analysis of the proposed system are detailed in Sect. 4.4. Section 4.5 concludes this paper and provides future directions.

4.2 Literature Review

In this section we will review the work done in the area of incident prediction in recent years. This includes the work done in traffic behavior modeling using big traffic data, approaches for prediction of traffic behavior and incident prediction using deep learning or other modeling techniques, and the work done by researchers to model the impact of incidents in terms of congestion, travel time variability, etc., by using different techniques.

An incident detection method using vehicles data collected by GPS is proposed in [9]. In this work, the vehicles data during the incidents has been collected by probe vehicles which are equipped by the GPS. They have proposed incident detection algorithms that are used for the prediction of incident by analyzing the collected traffic data during the incidents. The proposed algorithm is able to predict the spatial and temporal details about the traffic congestion that was caused due to the incident. Performance of the proposed method is measured by analyzing the vehicles flow in the outbound direction and a traffic flow simulator has been used for this purpose. The results presented in terms of incident detection rate and false alarm rate are better than the other methods, but, still, it has shortcomings that limit the scope and worthiness of the proposed approach. The authors have predicted the traffic congestion and have considered that it is in result of some incident on the road which is not true in general because incident is not the only cause of traffic congestion. Also the error rate calculated by using the defined formula is very high and sometimes more than 50%. The results have been compared with the simulation results instead of using the actual vehicles data on selected road network.

GPS data has also been used in [10] for the detection of traffic congestion and incidents. In this work, GPS data is collected from the driver's devices and the accessed GPS traces are used to identify the congestion on the road. Threshold values have been defied to categorize the traffic flow as congested or normal flow. This work uses a dataset of 24 incidents in total and simulation results have been used for analysis purpose. This could also be considered as a congestion prediction work that is capable of classifying the traffic into slow or very slow categories. Also, due to the very small dataset, where simulation results have been evaluated instead of using the real traffic dataset, traffic flow status results cannot be used as the key information so that it could be used to predict the occurrence of an incident.

In another work by Oskarbski et al. [11], telematics has been used for the prediction of incidents on the road intersections. They have discussed ad urban transport management system and have presented some simulation results obtained by selecting some features from the data and to detect the incidents on the road junctions which are equipped with the signals and working on the data that could lead them toward the incident prediction on the junctions and on the road networks.

In incident detection work, some methods have been proposed that are not directly related with the incident prediction but those could be used to analyze and predict the impact of incidents. Hojati et al. [30] present an approach to model the incident duration and its recovery time. Similarly, in [67], an approach has been

proposed to model the impact of incidents on travel time. Spatiotemporal impact of traffic incidents on road network has been predicted in [31]. A framework to estimate the variations in the travel time due to incidents has been proposed in [34]. So, a lot of work has been done in this area that deals with the incidents in terms of its impact on the road networks, travel time, etc. In [24], the authors have reviewed the methods that have been proposed to predict the traffic incidents duration by analyzing the incidents data.

In [21], the authors have proposed a dynamic Bayesian network approach to predict the crashes on highways in real-time by using the traffic speed conditions data. In this work, the relation between the crash incidents and the traffic states has been established, so that it could be used to predict the possibility of crashes on highways. Traffic states on the crash site have been divided into the upstream and downstream states. Here upstream is the state just before the crash and downstream is the state just after the crash in traffic flow direction. A vehicle speed threshold of 45 km/h was defined to identify the free flow (FF), i.e., traffic state is FF if the vehicles average speed is above 45 km/h. Average speed below 20 km/h identifies the jam flow (JF) and it is considered congested traffic (CT) if the flow is between the FF and JF threshold values. By using these three states values, nine combinations (upstream and downstream) have been defined to identify the occurrence of crashes in those states combinations. Crash reports data used in this work includes 551 records where 411 records were used for training and the remaining 140 records were used for the testing purpose. A confusion matrix was created to see the results. Several metrics based on the confusion matrix data were used to analyze the prediction results. Best DBN accuracy reported in this work is 76.4% where the false alarm rate reported in this case is 23.7%.

The authors in [68] have proposed a Bayesian structure equation model to predict the secondary incidents on the freeways. By analyzing the non-recurring congestion situations on the roads, their model identifies the road segments where there is the possibility of incidents. For this purpose, they use a boxplot and the area under that plot shows the area where there is traffic congestion and it could cause an incident. Bayesian neural networks (BNN) have been implemented and its results were compared with the logistic regression model. Hojati et al. in [67] have worked on the traffic incidents by modeling their impact on the travel time. They have used historic traffic data and nonrecurrent congestion has been identified by analyzing the vehicles speed data.

Traffic incidents detection work has done by using the data source other than the traffic data as well. Some researchers have used social media data, e.g., Twitter data, and by analyzing the tweets in a particular area, they have predicted the occurrence of road and other incidents [69–73]. In [59], the authors have presented an approach to detect events by analyzing Twitter data. In this work, they have collected Twitter data for a specific period of time (days) having some specific keywords. For example, they have collected Twitter data to analyze the traffic flow in the UK by using some traffic related keywords and by the number of tweets falling in that criteria. They have done this in some specific region to identify some traffic problems (e.g., high density). They have concluded that the increased number of

tweets regarding traffic gives insight of some traffic problems in that area. The work is further extended in [60]. In another work [61], a similar analysis approach has been adopted to detect the traffic conditions on the roads in Jeddah city.

4.3 Methodology

In this section, we will give details about the methodology used in this work. This includes detailed discussion about the input dataset and the deep learning models used to predict the incidents on the road network. First, we have given an overview of the PeMS incident dataset and the terms used in this data. In the next section, we have discussed in detail the datasets used in this work and the process to parse these datasets to be used as an input to the deep learning algorithm At the end, we have given details about the deep learning model used for the prediction purpose in this work. Algorithm 1 gives the overall work-flow of incident prediction work.

Algorithm 1 Incident detection main algorithm

Input: Vehicles and Incident Data.
Output: Predicted Incidents.

```
raw_inc_data ← load_incident_data
vds_list_all ← load_I5N_vds_data
vehicle_data ← load_vehicles_data
if incident_data is parsed then
    inc_prsd_dt ← load_parsed_data
    if deep_model is trained then
        pred_incidents ← MakePredictions(model_trained)
    else
        model_conf ← load_model_config()
        model_trained ← TrainModel(inc_prsd_dt, model_conf)
        pred_incidents ← MakePredictions(model_trained)
    end if
else
    inc_prsd_dt ← ParseIncidentsData(raw_inc_data, vds_list_all, vehicle_data)
    model_conf ← load_model_config()
    model_trained ← TrainModel(inc_prsd_dt, model_conf)
    pred_incidents ← MakePredictions(model_trained)
end if
return pred_incidents
```

4.3.1 Input Data Collection

In this section, we will give some details about the input incidents dataset which we have collected from PeMS [65]. For incident dataset in PeMS, the following terms have been used:

- CHP (California Highway Patrol)
- CAD (Computer-aided Dispatch)
- TASAS (Traffic Accident and Surveillance Analysis System)

CHP incident reports provide all the incident data found in CHP CAD (Fig. 4.1). On the other hand, TSAS includes all the accidents data on state highways which are manually verified by the Caltrans staff. Therefore, it approximately takes 1–2 years to report TSAS data to PeMS. TSAS records in PeMS provide the following details about the incidents:

- Starting time,
- Freeway,
- Direction,
- Postmile,
- Severity, and
- Location of incident.

Data available on the incidents is in the form of different graphs and reports. Reports on incidents data are available in different formats and provide different information. Following are some types of incident reports available in PeMS:

- Time Series, view incidents over a time range.
- Time of Week, radial (CHP incidents only), view incidents in a radial chart by time of day and day of week
- Time of Day, view incidents over time of day by hourly average
- Day of Week, view the total number of incidents for each day of week
- Duration (CHP incidents only), view distribution of the durations of incidents
- Characteristics (TASAS only), view percentage breakdown by selected characteristics for a time range
- Detail, view listing of individual incidents over a selected time range

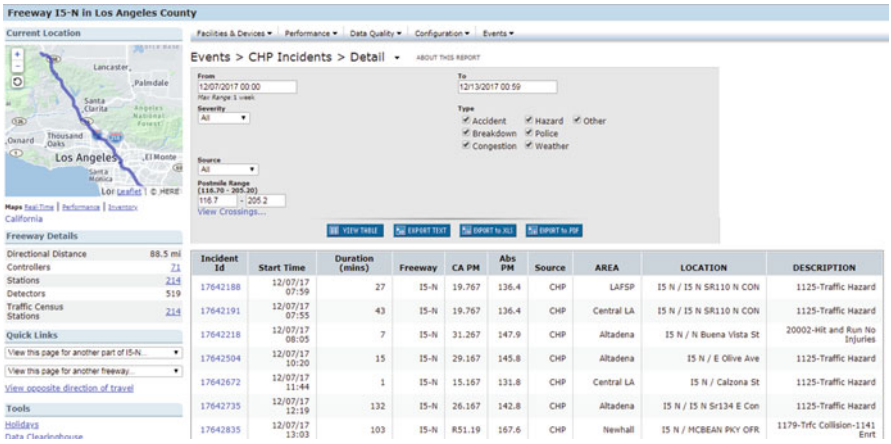


Fig. 4.1 CHP incidents data collection from PeMS on freeway I5-N in Los Angeles County

- Contours, view concentrations of incidents along a corridor segment over time period
- Comparison (CHP incidents only), view comparison of incident types with average duration
- Relationships, view the relationship between performance measures and the number of incidents
- Spatial Distribution (CHP incidents only), view the number of incidents along corridor segment
- Segments (TASAS only), view the number of incidents along corridor segment

4.3.2 Input Data Preparation

We are using incidents data collected from PeMS to predict incidents on the freeways. We have collected the incidents, vehicles flow/speed, etc., from a small patch of the freeway I5. The length of selected corridor is 13.784 miles and two directions on this patch are denoted by I5-N and I5-S. For I5-N, there are 26 vehicle detection stations (VDSs) to collect the data about the vehicles traveling on this route. On the opposite direction, i.e., I5-S, there are 25 such stations for data collection. In this work, we are considering only one side of this freeway patch, i.e., 13.784 miles of I5-N.

Now the total length of I5-N freeway is around 796.5 miles. We are considering the whole I5-N freeway because the incident data is available for the whole I5-N highway and the number of VDSs on the whole I5-N is 948.

In this work, we have prepared incidents input dataset by combining the data collected from three different datasets. This includes incidents dataset (Table 4.1), VDS stations details dataset (Table 4.2), and the vehicles flow/speed/occupancy

Table 4.1 Schema of incidents dataset

S.No	Attribute name	Description
1	Incident.Id	Numeric Id value of incident
2	Start.Time	Timestamp value of the incident. This includes data and time in 24 h format
3	Duration..mins.	Time in minutes that describes the duration of the incident
4	Freeway	Name of the freeway, e.g., I5-N
5	CA.PM	Caltran postmiles value for the location where the incident occurred
6	Abs.PM	Absolute postmiles value that gives location of the incident
7	Source	Source of incident information/data collection, e.g., CHP, TSAS, etc.
8	Area	Name of the area where the incident was reported
9	Location	Gives details of location, e.g., freeway name with street/bridge/boulevard
10	Description	Numeric value of the incident type along with the description, e.g., traffic collision, traffic hazard, etc.

Table 4.2 Schema of vehicles detection station dataset

S.No	Attribute name	Description
1	Fwy	Name of freeway, e.g., I5-N in this case
2	District	Numeric Id values of different districts through that the selected highway passes
3	Count	Name of county
4	City	City name
5	CA.PM	Caltran postmiles value for that VDS
6	Abs.PM	Absolute postmiles value for that VDS
7	Length	Length of freeway patch covered by this VDS
8	ID	Numeric Id value of VDS
9	Name	Name of VDS station
10	Lanes	Number of lanes on freeway at a VDS
11	Type	Type of freeway, e.g., mainline
12	Sensor.Type	Types of sensors used at a VDS (e.g., loop detector)
13	HOV	High occupancy vehicle
14	Distance	(Self-defined attribute) This defines the distance between the two consecutive VDS stations. It is calculated by using the absolute postmiles (Abs.PM). If x is the Abs.PM value for first VDS and y is the Abs.PM value for the second VDS, then distance = $y - x$, where $y > x$

(Table 4.3) dataset. In the following paragraphs we will discuss in detail why we need to combine these datasets and how we did it to get a final incidents dataset.

Incident dataset we are using in this work has ten attributes that provide different information about the incidents occurred on the freeway. The information about which we are concerned in our work includes the incident timestamp, its duration, exact location, and type of incident. We already have freeway information, because we are collecting the incident data for I5-N freeway. To get incident location, we are not using CA.PM because Abs.PM provides us the exact incident location by providing the distance in miles. Similarly, three other parameters (source, area, and location) are not very important while linking the incident data with the other freeway vehicles data. On the other hand, description attribute is very important that gives us the type of an incident. Therefore, we can say that the following five (out of ten) attribute values in our incident dataset are very important to combine this data with the other traffic data on the same freeway patch and at the same time.

- Incident Id
- Start time (time is further broken into minutes, hours, days, months, and years attribute values.)
- Duration minutes
- Abs.PM
- Description

Table 4.3 Schema of vehicles dataset

S.No	Attribute name	Description
1	Timestamp	Defines the time when data is captured at a vehicle detection station (VDS). Timestamp gives both date and time when data was calculated at a specific VDS
2	StationId	Id of a vehicle detection station (VDS). In this data, station Id is a numeric value
3	StationTotalFlow	Total number of vehicles passed through a specific VDS at a specific time interval
4	StationAvgOcc	Average occupancy rate; calculated at a VDS at a given time interval defined in timestamp attribute
5	StationAvgSpeed	Average speed calculated at a specific VDS at specific time interval
6	StationPercent Observed	Number of lanes at this VDS station
7	Lane1TotalFlow	Number of vehicles in Lane1
8	Lane1AvgOcc	Average occupancy calculated w.r.t. lane1
9	Lane1AvgSpeed	Average speed calculated at lane1
10	Lane1Observed	Either values observed or imputed for lane1
11	Lane2TotalFlow	Number of vehicles in Lane2
12	Lane2AvgOcc	Average occupancy calculated w.r.t. lane2
13	Lane2AvgSpeed	Average speed calculated at lane2
14	Lane2Observed	Either values observed or imputed for lane2
.	.	.
.	.	.
.	.	.
35	Lane8TotalFlow	Number of vehicles in Lane8
36	Lane8AvgOcc	Average occupancy calculated w.r.t. lane8
37	Lane8AvgSpeed	Average speed calculated at lane8
38	Lane8Observed	Either values observed or imputed for lane8

As we are using the incident dataset with the freeway traffic dataset, we will give details about the freeway traffic datasets as well. Freeways traffic dataset provides us the information about the vehicles flow/occupancy/speed, etc., at a given vehicle detection station on the selected freeway. Vehicle detection stations are the monitoring stations on the freeways that are equipped with different kind of devices (loop detectors, cameras, etc.) to monitor the traffic and to collect the traffic data. Suppose, if we are interested in collecting vehicles flow on the freeway at the time of the incident, then the data collected from these VDSs can provide us the following information:

- Station Id
- Timestamp
- 5-min interval vehicles data

As shown in Table 4.3, there are 38 input attributes but if we are talking about the vehicles flow only, then we are interested only in the flow values, VDS Id, and

timestamp values. Timestamp value is in the format "dd:mm:yyyy hh:mm," so we have broken it down to get minutes, hours, day, month, and year values. In addition to this, we can also get the day of the week values (Saturday–Friday) which are very important to see different traffic patterns on specific days, e.g., on weekends. Station Id gives us the numeric Id value of VDS on the freeway and 5 min flow values are collected from the total flow attribute values.

Now, we want to use both datasets to predict incidents on the freeways, and we need to combine both datasets so that it could be used to predict the incidents. For this purpose, we have to relate each incident with a VDS within the vicinity of that a particular incident was occurred. But in the incident dataset, we are not given the VDS Id. Instead we are given the absolute postmiles (Abs.PM) values that give the location of the incident on the freeway. On the other hand, the vehicles dataset provides the unique VDS Id for each vehicle detection station on the freeway. So, in order to relate the incidents dataset with the vehicles dataset, we have used VDS details (Table 4.2) that give details about the VDSs on the freeways.

VDS details dataset provides unique VDS Id (Station Id in vehicles dataset) and Abs.PM to define the exact location of the VDS. There are other attributes in this dataset but VDS Id (Station Id) and the Abs.PM are two attributes that could be used to combine the incident and the vehicles flow data. For this purpose, we first have calculated the distance between each successive VDS stations. We named this attribute "Distance" and added this as a 16th attribute in the VDS details dataset in Table 4.2. The purpose to calculate the distance between the two VDS stations on a freeway is that this gives the range or distance covered by a VDS on the freeway. And by using this information, we can identify the VDS station in that vicinity an incident was reported. If the incident was reported on x Abs.PM value, and the location of vehicle detection station v before this location is y, and the distance between this VDS station and next VDS station is d, then the reported incident is considered to be within the vicinity of VDS v if the condition in Eq. (4.1) is satisfied.

$$x \geq y \text{ and } x < y + d \tag{4.1}$$

Then we can say that the incident has occurred within the vicinity of that particulate VDS (whose Abs.PM location is y). So, we can link this incident data with the vehicles flow data collected at that VDS at the same timestamp values. By using this criteria, first of all, we have linked the incident data with the station Ids, i.e., each incident record was assigned the corresponding vehicle detection station Id and created a new dataset as defined in Table 4.4. This includes incident Id, year, month, day, hours, duration mins, description from the incident dataset (Table 4.1), and "StationId" is collected from the VDS dataset (Table 4.2). By using this new dataset, we have created a new dataset that not only contains the important attribute values from the incident dataset, but it can also combine the vehicles data attributes like flow, speed, occupancy, etc., as shown in Table 4.5. Algorithm 2 gives an overview of important steps to parse the input dataset and to get the resulting input dataset.

Table 4.4 Schema of incident+VDS dataset to link incidents with vehicles data

S.No	Attribute name	Description
1	Incident.Id	Numeric Id value of incident
2	Year	Year value (numeric), e.g., 2017
3	Month	Month of the year (numeric value), e.g., 11 is used to represent the month of November
4	Day	Day of the month (numeric value), e.g., 13
5	Hours	Hours of the day. These are also numeric values so these start from 0 to 23
6	Duration..mins.	Time in minutes that describes the duration of the incident
7	Description	Numeric value of the incident type along with the description, e.g., traffic collision, traffic hazard, etc.
8	StationId	Id of the vehicle detection station. Incident occurred within the vicinity of this VDS

Algorithm 2 Incident data parsing

Input: Raw Incident, VDS, and Vehicles Data.
Output: Parsed Incidents Dataset.

$vds\_list\_sorted \leftarrow SortVDSList(vds\_list\_all)$
$vds\_dist \leftarrow CalcDiff(VDS\_AbsPM_{i+1}, VDS\_AbsPM_i)$
$vds\_list\_dist \leftarrow AddDistAttribToVDS(vds\_dist)$
$vds\_in\_veh\_data \leftarrow UniqueVDS(vehicle\_data)$
$selected\_vds \leftarrow Get\ vds\_list\_dist\ value\ if\ exists\ in\ vds\_in\_veh\_data$
for $i = 1, i++, i \leq length(raw\_inc\_data)$ **do**
  for $v = 1, v++, v \leq length(vds\_list\_dist)$ **do**
    if $inc_i\ occured\ within\ the\ vicinity\ of\ vds_v$ **then**
      $inc\_vds\_data \leftarrow MergeIncVdsData(inc_i, vds_v)$
      $continue$
    end if
  end for
end for
$SplitIncTimeStampVals()$
$inc\_comb\_data \leftarrow Merge(vehicle\_data)$ {based on timestamp and stationIds}
if $Data\ veracity\ issues\ exist$ **then**
  $inc\_new\_data \leftarrow DealWithDataVeracityIssues()$ {e.g., remove NAs}
  $inc\_comb\_data \leftarrow inc\_new\_data$
end if
if $Required\ normalization\ is\ required$ **then**
  $inc\_norm\_data \leftarrow NormalizeData(inc\_new\_data)$
  $inc\_comb\_data \leftarrow inc\_norm\_data$
end if
$inc\_data\_final \leftarrow AddIncBasedOnIncDuration(inc\_comb\_data)$
return inc_data_final

Now each incident was linked with the corresponding vehicles data values. As we have combined each incident with the vehicles data by comparing some particular attributes values like hour, day, month, year, station Id, etc., each incident was linked with only one record in the vehicles dataset. There were two important attributes

Table 4.5 The final schema of incident+vehicles dataset used for incident detection

S.No	Attribute name	Description
1	stationId	This attribute defines the numeric value assigned to each vehicle detection station (VDS) on the highway. VDSs in PeMS are the data collection points on highways. Data used in this experiment includes I5-N highway and has 26 VDSs
2	dayOfMonth	Defines the day of a month in a Gregorian calendar in "dd" format. So, its value ranges from 1 to 31. This could be helpful in getting some traffic flow trends on some specific events, e.g., traffic flow on Christmas every year
3	Month	Gives the numeric value for a Gregorian month in "mm" format and its value ranges from 1 to 12
4	Year	Value for a Gregorian year in the "yyyy" format
5	Hours	Clock hours in numbers starting from 0 to 23. This could help us in identifying traffic flow trends and other information in specific hours in a day, e.g., traffic flow at 9 a.m.
6	weekDays	Day of a week, e.g., Monday. weekDays values are also in numeric format ranging from 1 to 7, where 1 represents Sunday, and 7 represents Saturday. This is important in identifying specific trends on specific days, e.g., on weekends
7–18	flow (flow_00, flow_05, flow_10, . . . , flow_55)	As defined on PeMS, flow defines the number of vehicles passing through a vehicle detection station (VDS) at a given time. In this dataset, we have used 5-min interval flow values where the flow_00 defines the total number of vehicles passing through a VDS during the first 5 min of an hour. Similarly, flow_55 defines the total number of vehicles passing through a VDS during the last 5 min of the hour
19	No_incident	This column is added to the data to define no incident. It contains binary data, i.e., 1 or 0. If no incident was reported during that hour, it is 1 else 0
20–38	Different incident types reported in this data have been used as attribute names. For example, Collision (with injury)	These attributes define the different types of incidents. These also contain binary values. 1 is used if an incident of that particular type is occurred, and therefore, the values of all the others incidents will be 0 and vice versa

values in the incidents dataset. One of them was timestamp that gives the temporal information about the incident and the other one is the duration of the incident. Duration of incident defines the duration in minutes of that incident. Both these are very critical and important in the sense that the incident's duration effects directly the vehicles flow, their speed, and occupancy on the incident location and a patch of road before and after it. For example, if a traffic collision was reported on 15:34, and its duration was 128 min, then it means that it affected the traffic till 17:42. But as we were comparing timestamp as well while combining an incident record with the

vehicles data, so only one record with an hour value 15 will indicate that incident and if there was no other incident during the next 2 h, then the next two records with hour values 16 and 17 will indicate no incident. But in actual, vehicles data, e.g., flow, has been affected during the next 2 h due to this incident. In addition to this, if an incident was occurred on 23:40 and continued for 50 min, then it means that it continued during the first hour of next day as well. Also, it may change the month and year values as well. Therefore, we have developed an algorithm (Algorithm 3) that manages the resulting incidents dataset and deals with these situations and relates multiple records in the vehicles dataset with an incident depending upon its start time and duration.

4.3.3 Deep Model for Prediction

We are using deep neural networks for prediction purpose in this work. In a neural network, many neurons are used in such a way that the output of a neuron could be used as an input to the other neurons in the network. Left most layer in the network is the input layer and the right most layer is the output layer. The number of neurons in input layer is the number of input parameters in our input dataset, whereas we are predicting traffic flow for a specific time interval, so the output layer returns one single neuron considered as an output or predicted value.

In this work we are using vehicles data (average occupancy) on a road network and some other parameters as input and training our deep model to predict the type of incident that could occur depending upon the input values provided for a particular record as an input. We have classified the incident into different types (almost 19) depending upon the incident type and the severity, duration, etc., of an incident. The architecture of our classification model is shown in Fig. 4.2. As shown in this figure, there are n input parameters to be fed to our deep model. Here the value of n depends upon the vehicles data attributes, e.g., speed, flow, occupancy, etc., used as input for incidents classification. There are k hidden layers in our deep model where the number of neurons in each hidden layer could be same in all or some of the hidden layers or it could be different from all or some of the hidden layers. Therefore, neurons in hidden layers in this figure have been represented by H_{ij}, where i represents the number of hidden layer and j represents the jth neuron in ith hidden layer. For example, H_{56} represents the 6th neuron in the 5th hidden layer in our deep classification model. Right most layer in our deep model architecture represents the output layer and it is showing the output values from O_1 to O_z where z represents the number of incidents classes defined in this work for classification. Training process using the deep learning model is given in Algorithm 4.

Algorithm 3 Algorithm to add incidents in records based on incident duration

Input: Incident input dataset.
Output: Incidents input dataset with additional incident records according to the incidents durations.

```
incDuration ← incInputData["durationMins"]
tmpIncData ← incInputData
for i = 1, i++, i ≤ length(incDuration) do
    if incDuration[i] has characters then
        getDurationInMinutes(incDuration[i])
    end if
end for
count ← 0
for r = 1, r++, r ≤ length(incInputData) do
    record ← incInputData[r]
    min ← r["minutes"]
    dur ← r["durationMins"]
    totalTime ← min + dur
    if totalTime > 59 then
        hours ← floor(totalTime/60)
        tmpMin ← 0
        tmpRec ← record
        while count ≤ hrs do
            if hours < 23 then
                tmpRec["hrs"] ← tmpRec["hrs"] + 1
            else
                tmpRec["hrs"] ← 0
                if tmpRec["day"] < 28 then
                    tmpRec["day"] ← tmpRec["day"] + 1
                else if tmpRec["mnth"] is 2 then
                    if (tmpRec["year"] is leapYear) && (tmpRec["day"] is 28) then
                        tmpRec["day"] ← tmpRec["day"] + 1
                    else
                        tmpRec["day"] ← 1
                        tmpRec["mnth"] ← 3
                    end if
                else if (tmpRec["mnth"] is 4|9|11) && (tmpRec["day"] < 30) then
                    tmpRec["day"] ← tmpRec["day"] + 1
                else if (tmpRec["mnth"] is 1|3|5|7|8|10|12) && (tmpRec["day"] < 31) then
                    tmpRec["day"] ← tmpRec["day"] + 1
                else
                    if tmpRec["mnth"] is 12 then
                        tmpRec["mnth"] ← 1
                        tmpRec["day"] ← 1
                        tmpRec["year"] ← tmpRec["year"]1
                    else
                        tmpRec["day"] ← 1
                        tmpRec["mnth"] ← tmpRec["mnth"] + 1
                    end if
                end if
            end if
            tmpIncData.add(tmpRec)
        end while
        count ← count + 1
    end if
end for
return tmpIncData
```

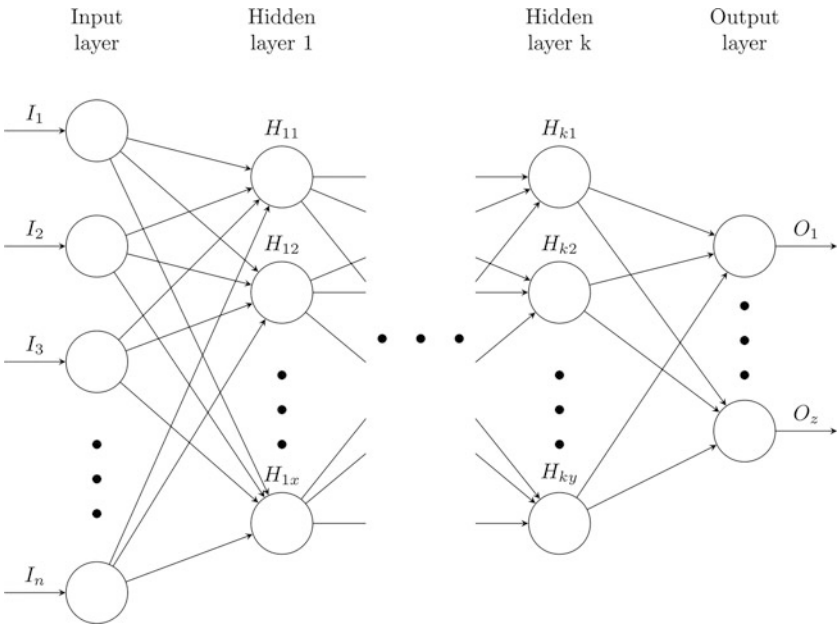


Fig. 4.2 Architecture of our deep classification model

Algorithm 4 Training incident detection deep model

Input: Parsed Incident Input Data.
Output: Trained Deep Model.

```
inc_data ← load_incident_data
batchSize ← setDefinedBatchSize()
epochs ← setNumEpochs()
nRepeatModel ← 5
count ← 1
while count ≤ nREpeatModel do
    model ← trainDeepModel()
    saveTrainedModel()
    count + +
end while
return model
```

4.4 Performance Analysis

This section defines our deep classification model configurations and the performance metrics which have been used to analyze the results obtained by this model.

4.4.1 Deep Model Setup

We have used the modified incidents data (Table 4.5) to predict the incidents in this work. For this purpose, we have used 3 months (September 2017–November 2017) vehicles occupancy and incidents data. There were 56,879 records in our input dataset. The data was divided in the ratio of 8, 1, 1, for training, testing, and prediction, respectively. We have executed our deep model with different configuration setups. The number of hidden layers and number of hidden units in those hidden layers were same in all the configuration setups. Also, because the same input data was used, the number of input parameters and output parameters was also same. The main difference was in the batch sizes and the number of iterations (epochs). In both the cases we used softmax activation function because we were trying to identify and then classify the incidents on the selected freeway I5-N.

We have used different combinations of vehicles data (flow/speed/occupancy) with the incidents data to predict the incidents on the freeways. For each resulting combination (e.g., vehicles flow and incident dataset as shown in Table 4.5) we have used different deep model configurations. Model configurations used in different sections are summarized in Table 4.6. Input data combination and model configuration details have been discussed in detail in respective sections for analysis of prediction results.

Table 4.6 Configuration of deep model for incident prediction using vehicles data

DL model	Description	Input parameters	Output parameters	Hidden layers	Hidden units	Batch size	Number of epochs	Activation function
Model configurations using vehicles flow								
1	Incidents prediction	18	1	5	18, 36, 36, 9, 3	100, 200, 500	500, 1000, 2000	Softmax
Model configurations using vehicles speed and flow								
2	Incidents prediction	30	1	6	30, 90, 45, 9, 3, 3	26, 52	500, 1000	Softmax
Model configurations using vehicles occupancy								
3	Incidents prediction	18	20	6	18, 18, 90, 90, 30, 20	26, 52, 100, 500	50, 100, 200, 500, 1000	Softmax
Model configurations using vehicles occupancy, speed, flow, and percent observed								
4	Incidents prediction	54	1	6	54, 108, 216, 54, 9, 3	13, 26, 52	100, 500	Softmax

4.4.2 Performance Metrics

We are using classification model for our deep learning model. Therefore, to calculate the accuracy of our prediction results, we have analyzed the results using the confusion matrix that has been created by comparing the predicted and the actual results. Different methods and techniques are used to evaluate the correctness of the predicted classification results. According to [74], it could be done by calculating the four terms defined below:

- True positives: Number of class instances that were identified correctly in prediction results.
- True negatives: Number of correctly identified class instances that do not belong to that class.
- False positive: Number of class instances that were incorrectly assigned to the class.
- False negative: Number of class instances that were not identified as an instance of a class.

The results obtained by using the above definition could be summarized in the form of a table known as a confusion matrix as shown below in Table 4.7.

We have calculated the average prediction accuracy by using the formula as given in Eq. (4.2).

$$\text{Average Accuracy} = \frac{\sum_{i=1}^{l} \frac{\text{tp}_i + \text{tn}_i}{\text{tp}_i + \text{fn}_i + \text{fp}_i + \text{tn}_i}}{l} \tag{4.2}$$

In the above formula, l is the number of classes in multi-class classification problem. tp_i represents the true positives, fn_i represents the false negatives, fp_i represents the false positives, and tn_i represents the true negatives for class C_i.

In addition to these classification results, we have used system generated mean absolute error (MAE), mean absolute percentage error (MAPE), and root mean squared error (RMSE) to analyze the prediction results. MAE is used to show the closeness between the actual and the predicted values and MAPE shows the relative difference between the actual and the predicted values. MAPE is not suitable to calculate error rate if the input data or actual values contain zeros because in this case it suffers from the division by zero error. RMSE is used to calculate the standard deviation of the prediction errors. It tells how much scattered are the residuals and thus tells how far are the predicted data points from the regression

Table 4.7 Confusion matrix for classification of predicted class instances

Classification categories	Classified as positive	Classified as negative
Positive	True positive (tp)	False negative (fn)
Negative	False positive (fp)	True negative (tn)

line. MAE, MAPE, and RMSE values are calculated by using the formulas given in Eqs. (4.3), (4.4), and (4.5), respectively.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|V_i - P_i| \tag{4.3}$$

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|V_i - P_i|}{V_i} \tag{4.4}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(V_i - P_i)^2}{N}} \tag{4.5}$$

Here N is the size (number of values predicted by the model) of dataset used for prediction purpose, V is the set of actual values used as labels, and P is the set of values predicted by our deep learning model.

4.4.3 Incident Prediction Using Vehicles Flow

In this section, we will discuss the prediction results that have been obtained while using vehicle flow with the incident dataset as shown in Table 4.5. Note that for this work, we have used the first 18 input parameters as an input to our model and instead of predicting incident classes we just have predicted whether an incident is reported or not. So, we have only one output variable instead of the 18 or more incident classes as shown in this table. Model configuration and input/output parameters details are given in Table 4.6 (DL Model 1). As shown in the table, we have executed our deep model with different batch sizes. But because of limited space, we are showing only the results obtained by executing the model with the batch size 500 as shown in Fig. 4.3. We have calculated MAE, MAPE, and RMSE for all the cases. The results show that we obtained very low MAE and RMSE error rate, but on the other hand, high MAPE values are pointing toward the low accuracy rate in prediction results which is true unfortunately. We have discussed these results in detail in discussion and analysis section (Sect. 4.4.7) and not only have highlighted the factors behind these results but also have shed light on how can we improve them.

4.4.4 Incident Prediction Using Vehicles Flow and Speed

In case of an incident on the road, it affects the average speed of vehicles and in addition to the decrease in vehicles flow, their average speed also decreases in that

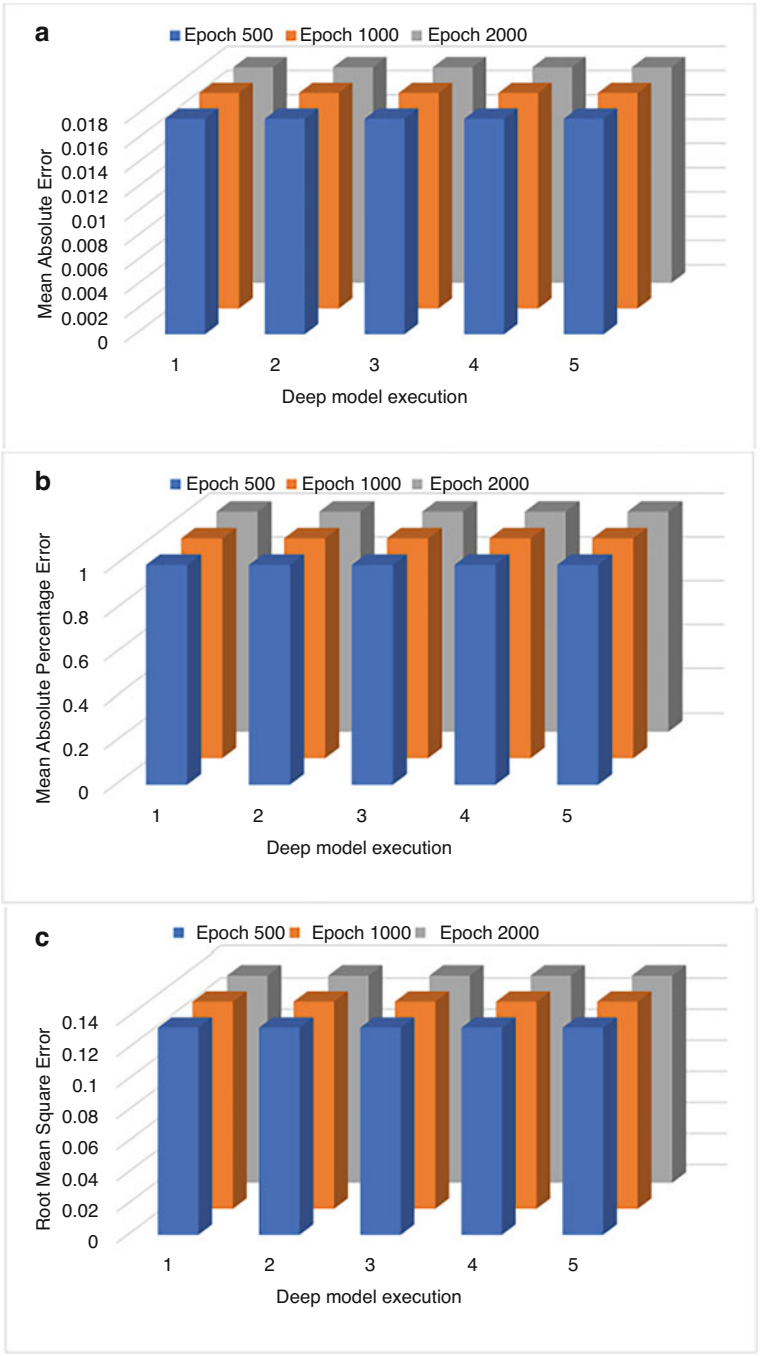


Fig. 4.3 (**a**) MAE, (**b**) MAPE, and (**c**) RMSE values when using vehicles flow and incident data with the batch size 500

Table 4.8 Description of additional speed attributes added to incident data to combine vehicles flow and speed for incident prediction

S.No	Attribute name	Description
19–30	speed (speed_00, speed_05, speed_10, ..., speed_55)	As defined on PeMS, speed defines the average speed of vehicles passing through a vehicle detection station (VDS). In this dataset, we have used 5-min interval speed values where the speed_00 defines the average vehicle speed calculated at a VDS during the first 5 min of an hour. Similarly, speed_55 defines the average speed value at a VDS during the last 5 min of the hour
31	IncidentOccured	A binary valued attribute to define whether an incident was reported during this time slot or not. 1 means a predicted incident (of any type) and 0 means no incident

place. Therefore, we can say that unusual change (decrease) in the average speed is also helpful in identifying the occurrence of an incident on the road. So, in this section, instead of just using the vehicles flow for incident prediction, we have used 5-min interval vehicles average speed values as well to predict the incident. Therefore, we have added 12 more input parameters in addition to the parameters shown in Table 4.5. The description of these additional input parameters is given in Table 4.8.

By adding 12 5-min interval speed values, now the number of input parameters in this case is 30, because, before we were using first 18 parameters (including 12 5-min interval flow values) in Table 4.5 as input parameters. Now the parameter numbered as 31 in Table 4.8 is the only output parameter and its expected values is defined in this table.

Table 4.6 (DL Model 2) gives the details about the configuration settings of the deep model used in this case for training and prediction. The results obtained by analyzing the predicted values are shown in Fig. 4.4. This shows the MAE, MAPE, and RMSE values when batch size was 26. Although we have executed the deep model by using different configuration setups, here we have shown only the results obtained by executing the model with batch size 26 because of similar trends shown in other results as well. By analyzing these graphs, we can say that these results also look similar to those discussed in the previous section where we used only the vehicles flow values with the incidents dataset. A detailed discussion on these similar trends and accuracy/error rate is given in Sect. 4.4.7.

4.4.5 Incident Prediction Using Vehicles Flow, Speed, Occupancy, and Percent Observed

Vehicles occupancy in PeMS data defines the amount of time a vehicle takes to pass through a VDS on the freeways. In order to see the effect of occupancy when used with other vehicles data attributes like flow and average speed, we have now used the dataset that includes 5-min interval values of all (flow, speed, and occupancy)

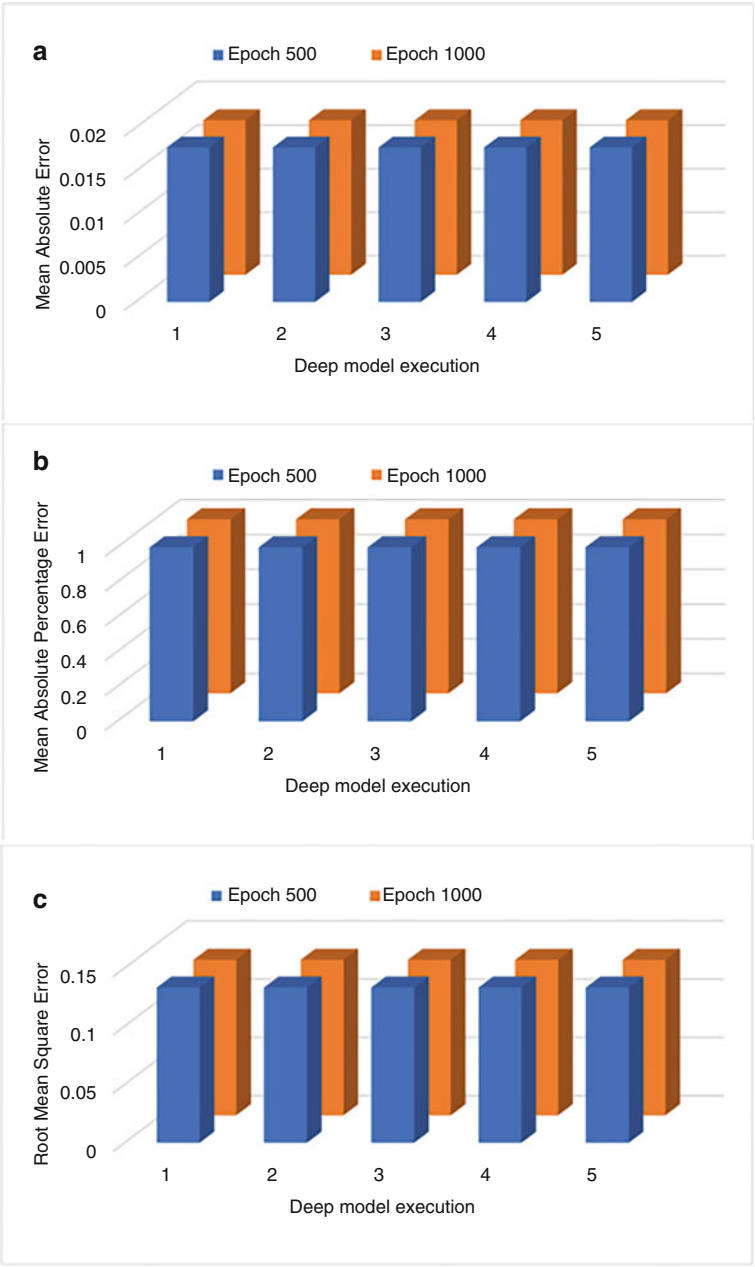


Fig. 4.4 (**a**) MAE, (**b**) MAPE, and (**c**) RMSE values when using vehicles speed with flow and incident data with the batch size 26

Table 4.9 Description of additional occupancy and percent observed attributes added to incident data prediction

S.No	Attribute name	Description
31–42	Occupancy (occupancy_00, occupancy_05, occupancy_10, …, occupancy_55)	As defined on PeMS, Occupancy is the amount of time it takes for a vehicle to pass through a VDS deployed on the freeways. In this dataset, we have used 5-min interval occupancy values where the occupancy_00 defines the average vehicle speed calculated at a VDS during the first 5 min of an hour. Similarly, occupancy_55 defines the average speed value at a VDS during the last 5 min of the hour
43–54	StationPercent Observed (obsrvd_00, obsrvd_05, obsrvd_10, … obsrvd_55)	The percent observed refers to the percentage of data points used to generate a report that were directly observed versus imputed. The higher the percent observed, the better the data quality of a report. The percent observed is summarized on every PeMS report and should always be referenced to understand the underlying data quality
55	IncidentOccured	A binary valued attribute to define whether an incident was reported during this time slot or not. 1 means a predicted incident (of any type) and 0 means no incident

these attributes. In addition to these attributes, we also have considered the number of lanes that were in use when the incident was occurred. The number of lanes at a VDS is defined by the attribute "percent observed." The details of additional 5-min interval attributes used in this section are given in Table 4.9.

Now we are using 12 5-min interval values for occupancy, and another 12 5-min interval values for percent observed as well. So, now the total number of input attribute values is 55 which includes first 18 attributes from Table 4.5, 19–30 attributes from Table 4.8, and the next 24 attributes (31–54) from Table 4.8, whereas the 55th attribute is used as an output. Complete input parameters details and configuration settings details are shown in Table 4.6 (DL Model 4). Prediction results obtained by executing this model are presented in Fig. 4.5. Again, instead of showing all the results obtained by executing our deep model with different configurations settings, we have shown the results for one model with batch size 26 only. The results show that although we have combined all the four vehicles traffic data attribute to improve the incident prediction accuracy, the graphs shown does not indicate the signs of any big change.

4.4.6 Incident Class Prediction Using Vehicles Occupancy Data

In the above sections, we have discussed the models and results where there was only one output attribute value. It was either predicting whether an incident has occurred or not. There was no classification of results according to the incidents category. In this section, we are discussing the model where we have categorized

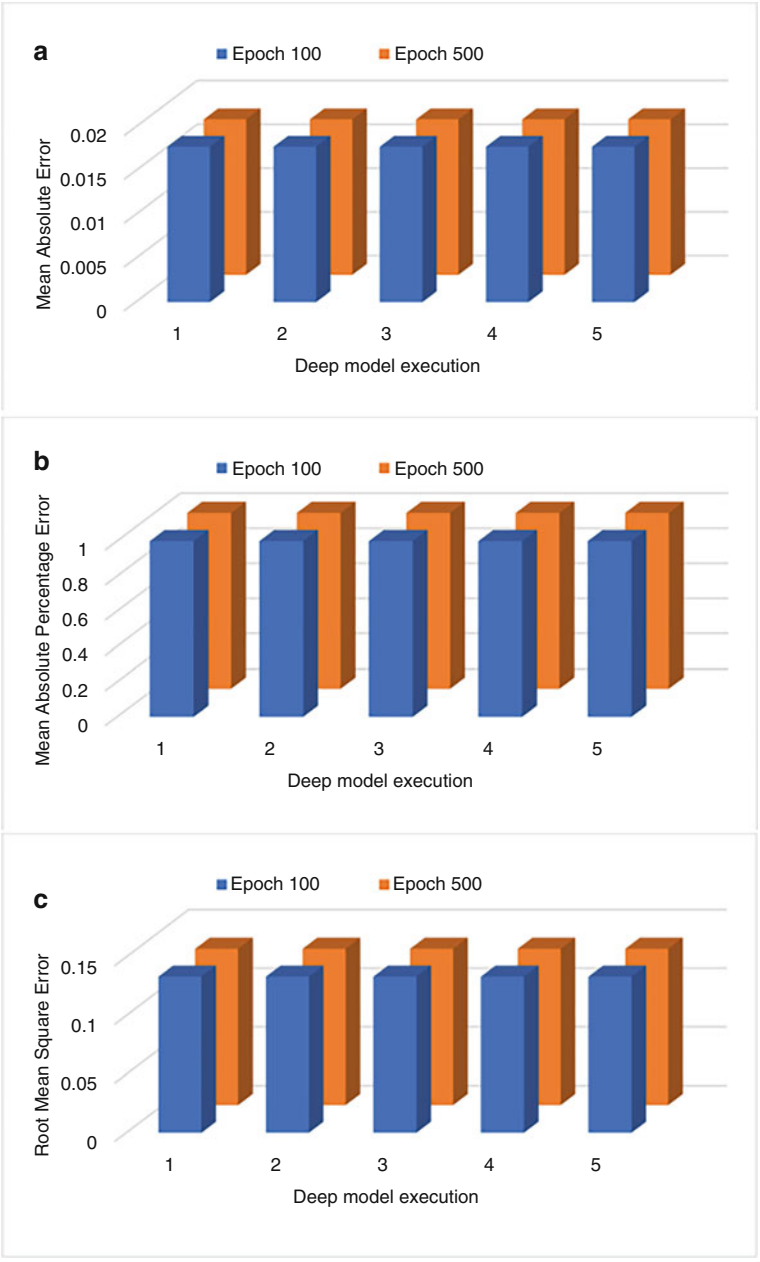


Fig. 4.5 (**a**) MAE, (**b**) MAPE, and (**c**) RMSE values when using vehicles occupancy, speed, percent observed with flow, and incident data with the batch size 26

the output according to their type. In the incident data which is used in this work, around 19 different types of the incidents have been reported. Therefore, here we are using multi-class classification model and the incidents for each record in our incident dataset have been represented by using the one-hot encoding notation. That is, we have created a matrix with 20 columns (19 incident classes plus one additional entry to represent no incident) and if a record in our dataset is reporting an incident of specific type, then the corresponding category value for that record is 1 and all the other values are 0. A very simple case to explain this thing is that if a record is not reporting an incident, then the column representing the "no_incident" for that record will contain 1, whereas all the other incident categories for that record will have 0.

We have used vehicles occupancy data in this work with the incident data for prediction and classification of incidents on the freeway. There are 18 input attributes where the first six attributes are same as shown in Table 4.5, and the rest 12 5-min attributes are same as shown with the "S.No" values 31–42 in Table 4.9. As we are predicting the class of expected incident as well, now the number of our output parameters is 20. Configuration settings and other details are given in Table 4.6 (DL Model 3).

4.4.7 Discussion and Analysis

In this work, we have used vehicles 5-min interval data in combination with the incidents data provided by PeMS [65] for incidents prediction on the selected patch of freeway I5-N. We used different combinations of vehicle data attributes with the incident input data to predict the incidents. In Sect. 4.4.3, we have used 5-min interval vehicles flow values and the prediction results obtained by analyzing the predicted and the actual values have been shown in Fig. 4.3. In this figure, we have shown the MAE, MAPE, and RMSE values obtained by executing the defined deep model with specific configuration settings. MAE and RMSE values in this figure were very low that shows the high accuracy of the predicted results but on the other hand, there is MAPE values graph as well which shows high error rate. We will discuss its reason in detail at the end after discussing all the results from the other sections as well. We are using the vehicles data with other attributes because it has direct link with the incidents on the freeway. Here we have used flow values because it is obvious that the vehicles flow values will be affected in case of an incident.

Another factor that is affected by an incident on the freeways is vehicles speed in that area. PeMS provides us 5-min interval vehicle speed data as well. So, taking the advantage of this data, we included these speed values as well to help our model to detect the anomalies in the data and to predict the incidents. The details about our deep model with the vehicles flow and average speed data as input were given in Sect. 4.4.4. Same like the previous model where we used only vehicles flow values, we executed our model defined in this section multiple times with the different configuration settings to see the difference between the output produced

by our deep models in both these sections. Although this model was executed with different configurations, here we have presented the results obtained by executing the model with the batch size 26 in Fig. 4.4. In this section, deep models were executed with two different batch sizes and two different epochs values as shown in Table 4.6 (model 2). The results in this case were almost same as we obtained in the case where we used only the flow values, i.e., MAE, and RMSE values were quite low but MAPE values were very high. We executed these models with many other configurations but in all the cases the results were almost the same. Therefore, we are including the results for only one configuration settings.

Another important factor when working with vehicles data is occupancy. Occupancy according to PeMS manual [65] is the time a vehicle takes on average while passing through the vehicle detection stations on the freeways. So it is obvious that there will be a change in its value on the places before and after the place where the incident was occurred. In addition to this it is very important that how the data was collected. In PeMS, this is named as "vehicle percent observed" that shows the percentage value to represent whether the results given were reported/recorded by the VDS or they were imputed. Therefore, we included these two attributes values in the new incident dataset as well. So, at this point the incident data was including the values of four vehicles data attributes (flow, speed, occupancy, and vehicle percent observed). Each of these attributes values were in 5-min interval data form, so by including the other attributes as well, there were 54 input parameters in our input dataset this time as discussed in Sect. 4.4.5. The results obtained in this section are shown in the graphs in Fig. 4.5. Although we included two other important vehicles data attributes in this dataset for incident prediction, the results obtained in this section were also similar to the results obtained in the previous sections.

From the results in the previous sections, we identified that there are some factors that are affecting the prediction accuracy and are critical in the sense that they should be addressed and issues related to them should be sorted out before using a dataset for prediction purpose.

In Sect. 4.4.6, we have used the input data that divides the reported incidents into incidents classes according to their types and description. Here, instead of just using one variable to define the occurrence or non-occurrence of an incident, a multi-class classification model has been followed. The results obtained by this model are shown in the form of a confusion matrix shown in Table 4.10. It is clear from the confusion matrix that although there were 19 incident classes in our training dataset, not all of those incident types were present in the dataset used as an input for prediction purpose. The incident classes were assigned alphabets from A to T as shown in Table 4.11, where A represents the first incident class and T represents that no incident was reported/predicted. From those classes, ten classes were included in the prediction dataset but the model was unable to detect any of those incidents and in all the cases it predicted only the non-occurrence of an incident. Although if we calculate the accuracy by using the defined average accuracy formula, it shows accuracy more than 97% which is because of the fact that there were more than 97% records where the input data was not reporting an incident. It means that our models were unable to predict the incidents accurately and we have investigated the reasons

Table 4.10 Confusion matrix for multi-class incident prediction results

	A	B	C	D	E	F	H	L	P	T
A	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0
T	25	41	46	4	1	7	2	1	12	5555

Table 4.11 Types of incidents in PeMS incident data

S.No	Incident class	Incident type
1	A	1183-Trfc Collision-Unkn Inj
2	B	1182-Trfc Collision-No Inj
3	C	1125-Traffic Hazard
4	D	20002-Hit and Run No Injuries
5	E	20001-Hit and Run w/Injuries
6	F	1179-Trfc Collision-1141 Enrt
7	G	SIG Alert
8	H	1179-Trfc Collision-1141Enrt
9	I	DOT-Request CalTrans Notify
10	J	BREAK-FSP Req Traffic Break
11	K	BREAK-Traffic Break
12	L	FIRE-Report of Fire
13	M	CLOSURE of a Road
14	N	23114-Object Flying From Veh
15	O	CFIRE-Car Fire
16	P	MZP-Assist CT with Maintenance
17	Q	1125A-Animal Hazard
18	R	ANIMAL-Live or Dead Animal
19	S	1166-Defective Traffic Signals
20	T	No_Incident

which led us to these results so that researchers should address them and take care of those facts when using these kind of datasets for incidents prediction in future.

If we divide our prediction dataset (5694 records) according to different incidents that were reported in that dataset, then we can see that in this dataset, the number of incidents was very less, and it was just making less than 2.5% of the whole dataset used for incident prediction. The number of different incidents reported in the incident dataset used for prediction purpose is shown in Fig. 4.6.

As shown in the figure, 5555 records out of 5694 records in the dataset say that there was no incident recorded and only 139 of those records were representing

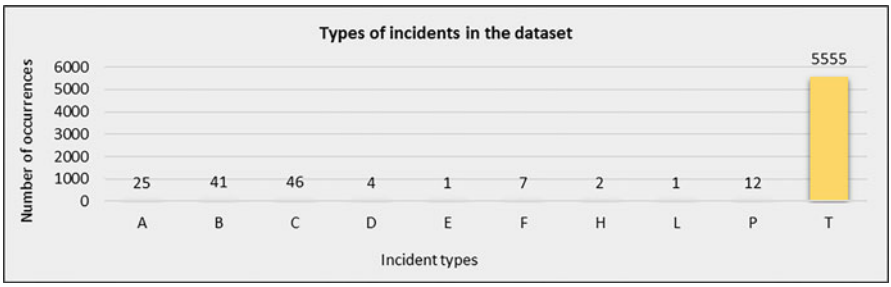


Fig. 4.6 Distribution of incident categories data in the prediction input dataset

an incident occurred on that selected patch of the freeway I-5N. This is the most important and probably the biggest factor that leads our model to predict no incident in almost all the cases. Because when more than 97.5% data is representing no incident, then it is very difficult to predict the rest 2.5% incidents accurately. If we see this scenario in the context of accuracy, then it has accuracy more than 97.5 where the model accurately predicted the non-occurrence of an incident, whereas only 2.5% results were not representing the actual incidents. But this accuracy, in actual, does not represent the correct picture of prediction results. In actual, the results obtained from this data are predicting the only value that was repeated in the dataset for more than 97.5%.

In our work, we have used vehicles flow, speed, and occupancy values in combination with the incidents data for the prediction of incidents. We have used this data for modeling because incidents on the freeways have direct link with these vehicles data attributes and it is obvious that in case of an incident the trend graphs of these values will reflect the change in their values. For the analysis of data to identify the change in the values of these attributes, we have selected the vehicles data during a week when an incident was reported in a specific area. On the selected patch of freeway I5-N, an incident was reported in the vicinity of the VDS station (Id = 715918), on 11th of September, 2017, after 11 a.m. The type of incident reported by the system was traffic collision, and its duration was 202 min. This shows that although no injury was reported, the collision was quite severe. It might have caused a congestion on that part of the road. Therefore, we have collected the vehicles for the whole week when the incident was reported.

In Fig. 4.7, we have shown the vehicles flow values during the week when the incident was occurred. It ranges from 10th to 16th September, 2017. Weekly vehicles flow values clearly show the general trends on weekends (Saturday and Sunday), the flow was very low in the morning but it was gradually increasing till 10 a.m. But on Monday, when the incident was reported it shows high flow during the morning peak hours, but it was then normal before the occurrence of the incident around 11:10 a.m. But if we see it carefully, we do not see any significant change in the graph or clear change in it before or after the incident.

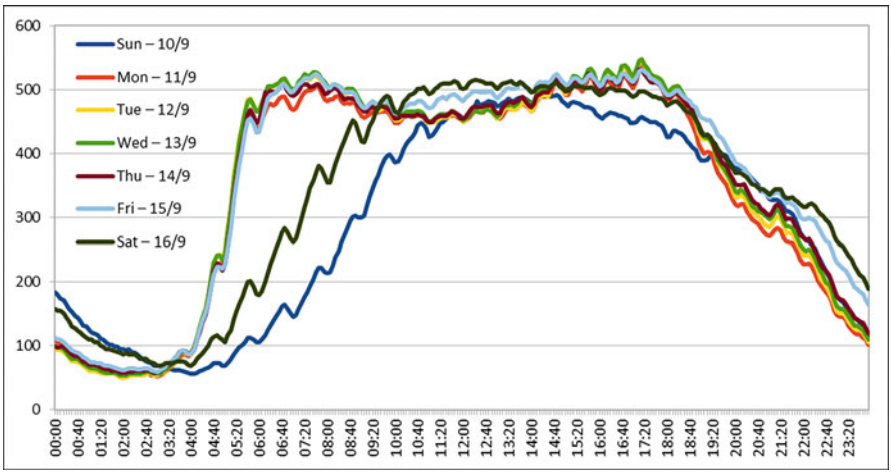


Fig. 4.7 Vehicles flow for the week 10–16 September 2017

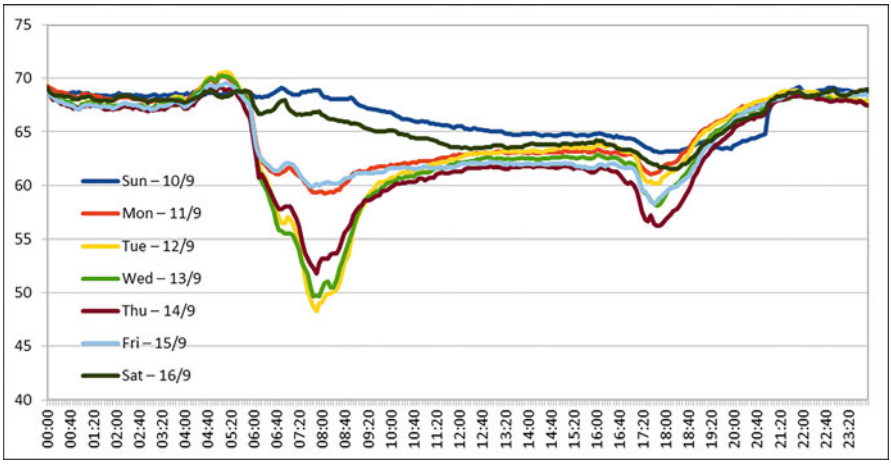


Fig. 4.8 Vehicles speed for the week 10–16 September 2017

Vehicles speed is also an important vehicles data attribute which is affected in case of an incident on a road. We also have examined the vehicles speed values during the week to see the trends before and after the same incident. Figure 4.8 shows the vehicles average 5-min interval speed values during the same week (10–16 September 2017). Vehicles speed graph will be showing high or normal speed values during the off-peak hours or morning hours till 10 a.m. on the weekends but it shows low speed values during the morning and evening peak hours in week days. It shows the normal trend during other times on week days as well. So, if we see the trend on Monday at 11 a.m. when this incident was occurred, same like flow, there is no significant change in the red line showing the speed values on Monday. This

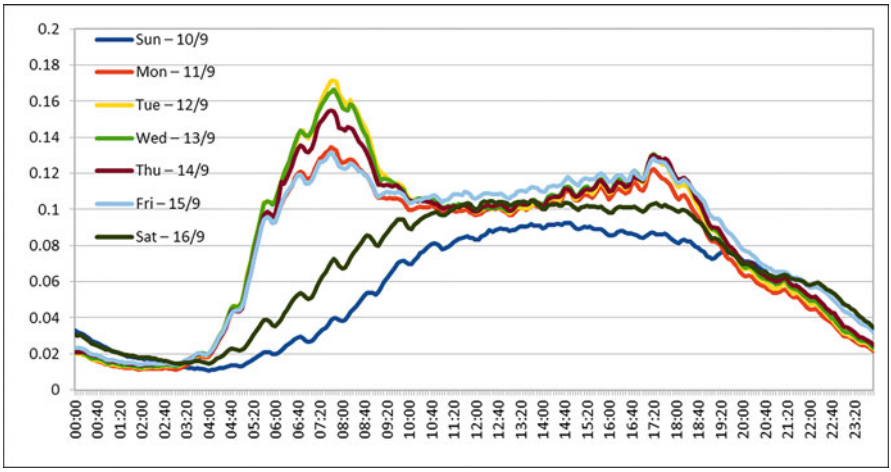


Fig. 4.9 Vehicles occupancy for the week 10–16 September 2017

also shows that the vehicles speed was not affected by the incident or there was no significant change which affected the vehicles speed.

In addition to the flow and speed values, vehicles occupancy values graph for selected week is shown in Fig. 4.9. During the peak hours, vehicles average speed decreases and in result their average occupancy value increases. This could be seen from the figure that occupancy values are very high during the week days peak hours especially in morning from 6 a.m. to 9 a.m. But it is very low especially on week days when there is very less traffic on the roads. Therefore, we expect that in case of an incident on the road, there could be a congestion and in result, occupancy value graph will show an increasing trend during that time. Occupancy graph here instead is not showing any significant change in the vehicles occupancy values.

We conclude from these facts that for incident detection using traffic data, it is not only important that the input data should be balanced but also we need to find the anomalies in the input dataset that could help our deep model to accurately predict an incident. There could be some abrupt changes in flow or speed or in other vehicle data attributes values. Also it is expected that there will be gradual normal change in the data after the incident. Although these changes could not be reflected due to many reasons as it happened in this case, there could be a number of reasons behind these facts discussed in the above paragraphs. One possible reason is that there would have been multiple lanes (may be four or five) in that area of the freeway and the collision was occurred on the right most lane which did not affect the flow. So, it is an important factor to know the number of lanes that were practically in use at the incident place at the time of incident. Another important factor is to know that how far was the vehicle detector station (VDS) from the place where the incident occurred. Because it is possible that the incident occurred in the blind area which is out of the range of the vehicle detection station because each VDS has an area

coverage range and as per our examination of VDS data, distance values between many VDS stations were bigger than their coverage length values. So, if this is the case, it might be possible that the traffic was affected in that area, but because it was outside the range of respective VDS, its effect was not very clear when traffic reached under the coverage area of next VDS. Data collection duration could also be a factor, as in this case we are using 5-min interval data and although it took long for the authorities to manage it completely, it might be possible that they were able to manage it in a way so that it affected the traffic for just few minutes and thus no change is reflected in the data. In addition to this, although we have compared the data for changes with the same station's data on the next day during the same hours, it could be more helpful if the data was covered with the same weekday's data, i.e., if the incident was recorded on Monday, then for comparison we should use the Monday's data during the same hours. Another important thing is that what were the circumstances that caused the incident. So, we can say that these are some of the factors that could help the researchers to make good incidents predictions in future.

4.5 Conclusion and Future Work

In this work we have used deep learning to predict incidents on the freeways using traffic data from PeMS. We have combined incident data with the traffic data attributes, e.g., flow, speed, and occupancy to predict incidents. Incidents on the road networks directly affect the vehicles data values, and therefore, we can train the deep model to learn from the vehicles data patterns to detect anomalies in the data and to predict the incident. We used different combinations of vehicle data attributes and incident dataset to predict the incidents. First, we combined 5-min interval vehicles flow and incident data, then we combined vehicles flow and average speed with the incident data, we also used 5-min interval average occupancy values with the incident dataset for prediction of incidents. We also used combination of four vehicle dataset attributes including flow, speed, occupancy, and station percent observed (48 5-min interval input attributes) for this purpose.

By using the input datasets, we executed our deep models with different configuration setups multiple times to see the variation in the results. We have presented all the results in the respective sections by using different performance metrics. Although the results were not quite satisfactory, and despite having very low error rates while using MAE and RMSE, we have no harm in saying that our deep models gained this accuracy by identifying the most recurring output class instead of identifying the correct incident class. But these results are still important for us and for other researchers as well because these highlighted some issues that are very important and should be considered while working with the incidents using deep learning. It is very important to use balanced input dataset for incident prediction and arranging such kind of data is not a piece of cake. With the data, we need to analyze the data patterns before and after the occurrence of an incident

both in spatial and temporal ways. Vehicles data pattern are also very important that could be analyzed on the similar weekdays/events, etc.

In future, first we would like to work on these data related issues. Because data is very important when using deep learning models. We will try to find some other datasets as well that could be helpful in dealing with the issues highlighted above and in addition to this we will see how deep learning could effectively be used to improve the accuracy of prediction results.

Acknowledgements The authors acknowledge with thanks the technical and financial support from the Deanship of Scientific Research (DSR) at the King Abdulaziz University (KAU), Jeddah, Saudi Arabia, under the grant number G-673-793-38. The work carried out in this paper is supported by the High Performance Computing Center at the King Abdulaziz University, Jeddah.

References

1. World Health Organization: Road Traffic Injuries. http://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries (2018). Accessed 27 Nov 2018
2. World Bank: The High Toll of Traffic Injuries: Unacceptable and Preventable. World Bank (2017). http://www.worldbank.org/en/programs/global-road-safety-facility/publication/the-high-toll-of-traffic-injuries-unacceptable-and-preventable
3. Cookson, G.: INRIX Global Traffic Scorecard. INRIX Research (2018). http://inrix.com/scorecard/
4. Schrank, D., Eisele, B., Lomax, T.: TTI's 2012 urban mobility report. Texas A&M Transportation Institute. The Texas A&M University System 4 (2012)
5. El Hatri, C., Boumhidi, J.: Traffic management model for vehicle re-routing and traffic light control based on multi-objective particle swarm optimization. Intell. Decis. Technol. **11**(2), 199–208 (2017)
6. Kim, H.J., Hoi-Kyun, C.: A comparative analysis of incident service time on urban freeways. IATSS Res. **25**(1), 62–72 (2001)
7. Skabardonis, A., Varaiya, P., Petty, K.: Measuring recurrent and nonrecurrent traffic congestion. Transp. Res. Rec. J. Transp. Res. Board **1856**(1), 118–124 (2003)
8. Ghosh, I., Savolainen, P.T., Gates, T.J.: Examination of factors affecting freeway incident clearance times: a comparison of the generalized f model and several alternative nested models. J. Adv. Transp. **48**(6), 471–485 (2014)
9. Asakura, Y., Kusakabe, T., Nguyen, L.X., Ushiki, T.: Incident detection methods using probe vehicles with on-board gps equipment. Transp. Res. C Emerg. Technol. **81**, 330–341 (2017)
10. D'Andrea, E., Marcelloni, F.: Detection of traffic congestion and incidents from gps trace analysis. Expert Syst. Appl. **73**, 43–56 (2017)
11. Oskarbski, J., Zawisza, M., Żarski, K.: Automatic incident detection at intersections with use of telematics. Transp. Res. Procedia **14**, 3466–3475 (2016)
12. Ayres, G., Mehmood, R.: On discovering road traffic information using virtual reality simulations. In: 11th International Conference on Computer Modelling and Simulation, 2009. UKSIM'09, pp. 411–416. IEEE, Piscataway (2009)
13. Mehmood, R.: Towards understanding intercity traffic interdependencies. In: Proceedings of the 14th World Congress on Intelligent Transport Systems (ITS), held Beijing, October 2007 (2007)
14. Zhao, Z., Chen, W., Wu, X., Chen, P.C., Liu, J.: LSTM network: a deep learning approach for short-term traffic forecast. IET Intell. Transp. Syst. **11**(2), 68–75 (2017)

15. Fouladgar, M., Parchami, M., Elmasri, R., Ghaderi, A.: Scalable deep traffic flow neural networks for urban traffic congestion prediction. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp. 2251–2258. IEEE, Piscataway (2017)
16. Lv, Y., Duan, Y., Kang, W., Li, Z., Wang, F.Y., et al.: Traffic flow prediction with big data: a deep learning approach. IEEE Trans. Intell. Transp. Syst. **16**(2), 865–873 (2015)
17. Jia, Y., Wu, J., Du, Y.: Traffic speed prediction using deep learning method. In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pp. 1217–1222. IEEE, Piscataway (2016)
18. Yu, R., Li, Y., Shahabi, C., Demiryurek, U., Liu, Y.: Deep learning: a generic approach for extreme condition traffic forecasting. In: Proceedings of the 2017 SIAM International Conference on Data Mining, pp. 777–785. SIAM, Philadelphia (2017)
19. Ma, X., Tao, Z., Wang, Y., Yu, H., Wang, Y.: Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transp. Res. C Emerg. Technol. **54**, 187–197 (2015)
20. El Hatri, C., Boumhidi, J.: Fuzzy deep learning based urban traffic incident detection. Cogn. Syst. Res. **50**, 206–213 (2018)
21. Sun, J., Sun, J.: A dynamic Bayesian network model for real-time crash prediction using traffic speed conditions data. Transp. Res. C Emerg. Technol. **54**, 176–186 (2015)
22. Ki, Y.K., Heo, N.W., Choi, J.W., Ahn, G.H., Park, K.S.: An incident detection algorithm using artificial neural networks and traffic information. In: Cybernetics & Informatics (K&I), 2018, pp. 1–5. IEEE, Piscataway (2018)
23. Agarwal, S., Kachroo, P., Regentova, E.: A hybrid model using logistic regression and wavelet transformation to detect traffic incidents. IATSS Res. **40**(1), 56–63 (2016)
24. Li, R., Pereira, F.C., Ben-Akiva, M.E.: Overview of traffic incident duration analysis and prediction. Eur. Transp. Res. Rev. **10**(2), 22 (2018)
25. Boyles, S., Fajardo, D., Waller, S.T.: A naive bayesian classifier for incident duration prediction. In: 86th Annual Meeting of the Transportation Research Board, Washington, DC, Citeseer (2007)
26. Nam, D., Mannering, F.: An exploratory hazard-based analysis of highway incident duration. Transp. Res. A Policy Pract. **34**(2), 85–102 (2000)
27. Lee, J.Y., Chung, J.H., Son, B.: Incident clearance time analysis for Korean freeways using structural equation model. In: Proceedings of the Eastern Asia Society for Transportation Studies (The 8th International Conference of Eastern Asia Society for Transportation Studies, 2009), vol. 7, pp. 360–360. Eastern Asia Society for Transportation Studies, Tokyo (2009)
28. Zhan, C., Gan, A., Hadi, M.: Prediction of lane clearance time of freeway incidents using the m5p tree algorithm. IEEE Trans. Intell. Transp. Syst. **12**(4), 1549–1557 (2011)
29. Vlahogianni, E.I., Karlaftis, M.G.: Fuzzy-entropy neural network freeway incident duration modeling with single and competing uncertainties. Comput. Aided Civ. Infrastruct. Eng. **28**(6), 420–433 (2013)
30. Hojati, A.T., Ferreira, L., Washington, S., Charles, P., Shobeirinejad, A.: Modelling total duration of traffic incidents including incident detection and recovery time. Accid. Anal. Prev. **71**, 296–305 (2014)
31. Pan, B., Demiryurek, U., Shahabi, C., Gupta, C.: Forecasting spatiotemporal impact of traffic incidents on road networks. In: 2013 IEEE 13th International Conference on Data Mining (ICDM), pp. 587–596. IEEE, Piscataway (2013)
32. Miller, M., Gupta, C.: Mining traffic incidents to forecast impact. In: Proceedings of the ACM SIGKDD International Workshop on Urban Computing, pp. 33–40. ACM, New York (2012)
33. Chung, Y., Recker, W.W.: A methodological approach for estimating temporal and spatial extent of delays caused by freeway accidents. IEEE Trans. Intell. Transp. Syst. **13**(3), 1454–1461 (2012)
34. Javid, R.J., Javid, R.J.: A framework for travel time variability analysis using urban traffic incident data. IATSS Res. **42**(1), 30–38 (2018)

35. Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.): Smart Societies, Infrastructure, Technologies and Applications. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST), vol. 224. Springer International Publishing, Cham (2018)
36. Tawalbeh, L., Basalamah, A., Mehmood, R., Tawalbeh, H.: Greener and smarter phones for future cities: characterizing the impact of gps signal strength on power consumption. IEEE Access **4**, 858–868 (2016)
37. Mehmood, R., Alam, F., Albogami, N.N., Katib, I., Albeshri, A., Altowaijri, S.M.: UTiLearn: a personalised ubiquitous teaching and learning system for smart societies. IEEE Access **5**, 2615–2635 (2017)
38. Muhammed, T., Mehmood, R., Albeshri, A., Katib, I.: UbeHealth: a personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities. IEEE Access **6**, 32258–32285 (2018)
39. Büscher, M., Coulton, P., Efstratiou, C., Gellersen, H., Hemment, D., Mehmood, R., Sangiorgi, D.: Intelligent mobility systems: some socio-technical challenges and opportunities. In: International Conference on Communications Infrastructure. Systems and Applications in Europe, pp. 140–152. Springer, Berlin (2009)
40. Mehmood, R., Graham, G.: Big data logistics: a health-care transport capacity sharing model. Procedia Comput. Sci. **64**, 1107–1114 (2015). Elsevier
41. Arfat, Y., Mehmood, R., Albeshri, A.: Parallel shortest path graph computations of united states road network data on apache spark. In: International Conference on Smart Cities, Infrastructure, Technologies and Applications, pp. 323–336. Springer, Cham (2017)
42. Mehmood, R., Meriton, R., Graham, G., Hennelly, P., Kumar, M.: Exploring the influence of big data on city transport operations: a Markovian approach. Int. J. Oper. Prod. Manag. **37**(1), 75–104 (2017)
43. Mehmood, R., Lu, J.A.: Computational Markovian analysis of large systems. J. Manuf. Technol. Manag. **22**(6), 804–817 (2011)
44. Aqib, M., Mehmood, R., Albeshri, A., Alzahrani, A.: Disaster management in smart cities by forecasting traffic plan using deep learning and gpus. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) International Conference on Smart Cities, Infrastructure, Technologies and Applications (SCITA 2017): Smart Societies, Infrastructure, Technologies and Applications, pp. 139–154. Springer International Publishing, Cham (2018)
45. Alazawi, Z., Altowaijri, S., Mehmood, R., Abdljabar, M.B.: Intelligent disaster management system based on cloud-enabled vehicular networks. In: ITS Telecommunications (ITST), 2011 11th International Conference on, pp. 361–368. IEEE, Piscataway (2011)
46. Alazawi, Z., Abdljabar, M.B., Altowaijri, S., Vegni, A.M., Mehmood, R.: Icdms: an intelligent cloud based disaster management system for vehicular networks. In: International Workshop on Communication Technologies for Vehicles, pp. 40–56. Springer, Berlin (2012)
47. Alazawi, Z., Alani, O., Abdljabar, M.B., Altowaijri, S., Mehmood, R.: A smart disaster management system for future cities. In: Proceedings of the 2014 ACM International Workshop on Wireless and Mobile Technologies for Smart Cities, pp. 1–10. ACM, New York (2014)
48. Alazawi, Z., Alani, O., Abdljabar, M.B., Mehmood, R.: An intelligent disaster management system based evacuation strategies. In: Communication Systems, Networks & Digital Signal Processing (CSNDSP), 2014 9th International Symposium on, pp. 673–678. IEEE, Piscataway (2014)
49. Arfat, Y., Aqib, M., Mehmood, R., Albeshri, A., Katib, I., Albogami, N., Alzahrani, A.: Enabling smarter societies through mobile big data fogs and clouds. Procedia Comput. Sci. **109**, 1128–1133 (2017)
50. Graham, G., Mehmood, R., Coles, E.: Exploring future cityscapes through urban logistics prototyping: a technical viewpoint. Supply Chain Manag. **20**(3), 341–352 (2015)
51. Mehmood, R., Nekovee, M.: Vehicular ad hoc and grid networks: discussion, design and evaluation. In: Proceedings of the 14th World Congress on Intelligent Transport Systems (ITS), held Beijing, October 2007 (2007)

52. Gillani, S., Shahzad, F., Qayyum, A., Mehmood, R.: A survey on security in vehicular ad hoc networks. In: International Workshop on Communication Technologies for Vehicles, pp. 59–74. Springer, Berlin (2013)
53. Alvi, A., Greaves, D., Mehmood, R.: Intra-vehicular verification and control: a two-pronged approach. In: 2010 7th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP 2010), pp. 401–405. IEEE, Piscataway (2010)
54. Nabi, Z., Alvi, A., Mehmood, R.: Towards standardization of in-car sensors. In: International Workshop on Communication Technologies for Vehicles, pp. 216–223. Springer, Berlin (2011)
55. Alam, F., Mehmood, R., Katib, I.: D2TFRS: an object recognition method for autonomous vehicles based on RGB and spatial values of pixels. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, vol. 224, pp. 155–168. Springer, Cham (Nov 2018)
56. Schlingensiepen, J., Mehmood, R., Nemtanu, F.C., Niculescu, M.: Increasing sustainability of road transport in European cities and metropolitan areas by facilitating autonomic road transport systems (ARTS). In: Wellnitz, J., Subic, A., Trufin, R. (eds.) Sustainable Automotive Technologies 2013, pp. 201–210. Springer International Publishing, Ingolstadt (2014)
57. Schlingensiepen, J., Nemtanu, F., Mehmood, R., McCluskey, L.: Autonomic transport management systems-enabler for smart cities, personalized medicine, participation and industry grid/industry 4.0. In: Intelligent Transportation Systems–Problems and Perspectives, pp. 3–35. Springer, Cham (2016)
58. Schlingensiepen, J., Mehmood, R., Nemtanu, F.C.: Framework for an autonomic transport system in smart cities. Cybern. Inf. Technol. **15**(5), 50–62 (2015)
59. Suma, S., Mehmood, R., Albugami, N., Katib, I., Albeshri, A.: Enabling next generation logistics and planning for smarter societies. Procedia Comput. Sci. **109**, 1122–1127 (2017)
60. Suma, S., Mehmood, R., Albeshri, A.: Automatic event detection in smart cities using big data analytics. In: International Conference on Smart Cities, Infrastructure, Technologies and Applications, pp. 111–122. Springer, Cham (2017)
61. Alomari, E., Mehmood, R.: Analysis of tweets in Arabic language for detection of road traffic conditions. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, vol. 224, pp. 98–110. Springer, Cham (Nov 2018)
62. Mehmood, R., Faisal, M.A., Altowaijri, S.: Future networked healthcare systems: a review and case study. In: Boucadair, M., Jacquenet, C. (eds.) Handbook of Research on Redesigning the Future of Internet Architectures, pp. 531–558. IGI Global, Hershey, PA (2015)
63. Usman, S., Mehmood, R., Katib, I.: Big data and HPC convergence: the cutting edge and outlook. In: International Conference on Smart Cities, Infrastructure, Technologies and Applications (SCITA 2017). Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, vol. 224, pp. 11–26. Springer, Cham (Nov 2018)
64. Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., Lew, M.S.: Deep learning for visual understanding: a review. Neurocomputing **187**, 27–48 (2016)
65. Berkeley, U.: Caltrans (2005) freeway performance measurement system (PEMS) 5.4. pems. eecs. berkeley. edu/Public (2005). Accessed 30 June 2005
66. Aqib, M., Mehmood, R., Alzahrani, A., Albeshri, A.P: A smart disaster management system for future cities using deep learning, GPUs, and in-memory computing. In: Mehmood, R., See, S., Katib, I., Chlamtac, I. (eds.) Smart Infrastructure and Applications: Foundations for Smarter Cities and Societies. Springer (2019). https://doi.org/10.1007/978-3-030-13705-2_7
67. Hojati, A.T., Ferreira, L., Washington, S., Charles, P., Shobeirinejad, A.: Modelling the impact of traffic incidents on travel time reliability. Transp. Res. C Emerg. Technol. **65**, 49–60 (2016)
68. Park, H., Haghani, A.: Real-time prediction of secondary incident occurrences using vehicle probe data. Transp. Res. C Emerg. Technol. **70**, 69–85 (2016)
69. Paule, J.D.G., Sun, Y., Moshfeghi, Y.: On fine-grained geolocalisation of tweets and real-time traffic incident detection. Inf. Process. Manag. **56**, 1119–1132 (2018)

70. Zhang, Z., He, Q., Gao, J., Ni, M.: A deep learning approach for detecting traffic accidents from social media data. Transp. Res. C Emerg. Technol. **86**, 580–596 (2018)
71. Gu, Y., Qian, Z.S., Chen, F.: From twitter to detector: real-time traffic incident detection using social media data. Transp. Res. C Emerg. Technol. **67**, 321–342 (2016)
72. Gutiérrez, C., Figueiras, P., Oliveira, P., Costa, R., Jardim-Goncalves, R.: An approach for detecting traffic events using social media. In: Emerging Trends and Advanced Technologies for Computational Intelligence, pp. 61–81. Springer, Cham (2016)
73. Nguyen, H., Liu, W., Rivera, P., Chen, F.: Trafficwatch: real-time traffic incident detection and monitoring using social media. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 540–551. Springer, Cham (2016)
74. Sokolova, M., Lapalme, G.: A systematic analysis of performance measures for classification tasks. Inf. Process. Manag. **45**(4), 427–437 (2009)

Chapter 5
Hybrid Statistical and Machine Learning Methods for Road Traffic Prediction: A Review and Tutorial

Bdoor Alsolami, Rashid Mehmood, and Aiiad Albeshri

5.1 Introduction

Mobility is one of the major dimensions of smart city design and development. Many approaches have been proposed to address smart mobility–related challenges [1]—for example, social media–based approaches [2–4], location-based services [5, 6], telematics [7], modeling and simulation–based approaches [8, 9], approaches based on vehicular networks (VANETs) and systems [10–12], autonomic mobility management [13–15], autonomous driving [16], mobility in emergency situations [17–22], approaches to improve urban logistics [2, 23–25], and big data–based approaches [2–4, 26, 27]. A recent book has covered a number of topics related to smart cities, including smart mobility [28].

Traffic flow modeling and prediction play important roles in smart city transportation systems. The modeling of transportation traffic is usually done by using data acquired through various sensors [10], such as inductive loops and Motorway Incident Detection and Automatic Signalling (MIDAS) [9], use of surveys [17], vehicular ad hoc networks [10], and social networks [2–4]. Various methods are in practice to model and predict traffic, including mathematical modeling [9, 17, 18], simulations [8, 19–21], and deep learning [22]. Accurate prediction of the transport network state can improve information services for travelers and help them

B. Alsolami (✉) · A. Albeshri
Department of Computer Science, FCIT, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: balsolami0069@stu.kau.edu.sa; aaalbeshri@kau.edu.sa

R. Mehmood
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: RMehmood@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_5

to make informed travel decisions. Furthermore, precise prediction of road traffic can improve road safety by decreasing congestion problems, air pollution, traffic costs, and accidents [29]. The predicted information leads to good planning of the traffic infrastructure.

Today, the transportation sector has truly entered the big data era. The rapid increase of Global Positioning System (GPS) device use in vehicles and on smartphones has provided opportunities for researchers to use transport data for studying traffic states and solving traffic problems. In addition, social networking applications such as Twitter and Facebook have become sources of traffic data because most people share their various statuses and environmental conditions, including the status of road traffic. Many governments make their transport data available so that researchers can analyze it and propose solutions to traffic problems [30].

In the last few years, many research attempts have emerged to provide accurate and timely traffic flow prediction models. However, most of the existing prediction models are based only on a single prediction method, such as statistical methods or machine learning methods. Neither statistical nor machine learning methods can completely capture traffic flow patterns, because of the complex relationship of traffic data. Statistical methods provide good performance when traffic data have a linear relationship, while machine learning methods work well with nonlinear traffic data [31]. Furthermore, most of the existing prediction models are built on stand-alone models because of the data and compute-intensive nature of the complex models. There is a need for novel prediction methods that provide higher accuracy for prediction of traffic with diverse characteristics. Moreover, there is a need to use distributed and parallel big data platforms for traffic prediction [32]. This chapter:

- Gives a review of traffic flow prediction and modeling methods.
- Discusses the limitation of each method.
- Introduces a review of various types of traffic data sources.
- Describes notable big data analysis tools.
- Describes a hybrid method for road traffic prediction and provides a tutorial on the process of hybrid traffic flow prediction. This method is based on the autoregressive integrated moving average (ARIMA) and support vector machine (SVM) methods.

The rest of the chapter is organized as follows. Section 5.2 reviews methods for road traffic analysis and prediction, and discusses a classification for the methods. We also discuss the limitation of using a single method type. In Sect. 5.3, we discuss a classification of the available traffic data sources. Big data analysis tools are introduced in Sect. 5.4. Section 5.5 gives a tutorial on the process of hybrid traffic flow prediction. In Sect. 5.6, we provide our conclusions on the chapter.

5.2 Traffic Flow Prediction and Modeling Methods

Road traffic modeling and prediction are important issues that face both individuals and governments because of increases in traffic congestion, accidents, and air pollution [33]. Because of their importance, many researchers have published several methods for traffic flow prediction. In general, and from the academic literature, we can categorize the existing methods into two types: long-term prediction and short-term prediction. Long-term prediction accuracy is affected by external factors such as weather conditions, road construction, and changes in road infrastructure. For these reasons, long-term prediction is not widely used. Conversely, short-term prediction has been widely used for road traffic prediction. Many methods have been proposed for short-term traffic prediction. As shown in Fig. 5.1, short-term prediction can be categorized into three types: statistically based methods, machine learning methods, and hybrid methods.

5.2.1 Statistical Methods

In this section, we discuss Kalman filtering (KF) and ARIMA methods.

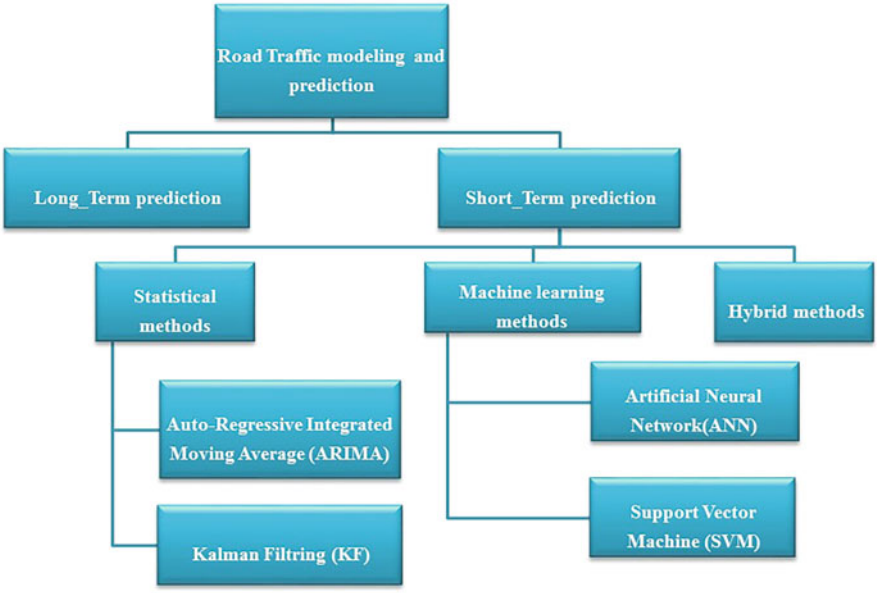


Fig. 5.1 Classification of traffic flow prediction methods

Kalman Filtering

Kalman filtering is a statistical method involving an algorithm that uses a set of measurements observed over time, which contain noise, and predicted outputs of unknown variables. In a KF approach, the estimated future state is based only on the estimated state of the previous time step [34]. Wang et al. [35] carried out research on the possibility of using KF for traffic flow prediction. Jinxing et al. [36] used a KF method to remove errors and redundancy from data sets. They concluded that filtering can increase the accuracy of the prediction model. Ahmad et al. [37] used social network traffic data (Twitter data) with a KF method for arrival time prediction. The results showed that the KF model has the ability to forecast vehicle arrival time with reasonable accuracy.

Autoregressive Integrated Moving Average

An ARIMA model is a tool for predicting the future values of time series [38]. Since it can represent traffic flow by a time series, we can fit an ARIMA model to predict future traffic flow. ARIMA (p, d, q) has been used for prediction by many researchers in many sectors, such as economics and transportation. In an ARIMA model the training data preprocessing to exclude the error data. After that, the data are classified into multiple data sets based on multiple time periods, and predictions are made through extraction of the correlation between sequences [39]. An ARIMA model is based on linear analysis [40]. Wang et al. [40] used ARIMA for short-term traffic flow prediction (see Sect. 5.2.4), and their ARIMA model provided good accuracy. Because the pattern of traffic flow appears as a seasonal pattern according to peak and off-peak traffic conditions, seasonal ARIMA (SARIMA) models are very suitable for modeling traffic flow behavior [41]. Kumar et al. [41] used 3 days of data to forecast the flow for the next day. The advantage of this type of model is the limited data set used for prediction.

5.2.2 Machine Learning Methods

Artificial Neural Networks

Artificial neural networks (ANNs) are the most widely used type of algorithm for traffic flow prediction. ANNs are a set of statistical learning algorithms. They have the ability to deal with complex problems and with missing and noisy data [42, 43]. ANNs contain multiple layers; the most widely used model is a multilayer perceptron (MLP). Changqing et al. [43] presented a classification and prediction framework for taxi hailing. They used K-means clustering to divide the data set into several clusters and used a neural network based on the clustering result to generate the prediction result. Florido et al. [44] used an NN algorithm for

congestion prediction in road networks. Achieving an accurate prediction result with a minimum value of square error (MSE) is the objective of an ANN. To achieve this aim, many researchers have developed a number of algorithms, such as the back propagation neural networks (BPNNs) proposed by Park and Rilett [45]. Pamuła [46] conducted research on analysis of traffic flow data using BPNNs. The traffic flow was represented by four classes of time series. The results showed the capability of neural networks to be used in intelligent transportation systems (ITSs). In ANNs, there are different processing elements called neurons; each of them takes multiple inputs and, on the basis of an internal weighting, only one output is produced. The neurons are organized into layers [39]. Ban et al. [47] used new neural networks named extreme learning machines (ELMs) to predict traffic states. The algorithm provided good performance in comparison with other prediction algorithms. However, time is the cost of this algorithm. ANNs need parallel architecture to process large data volumes in small amounts of time, and the cost of using ANNs is the large number of training data sets, which need large amounts of storage.

Support Vector Machines

An SVM comes under the statistical learning algorithm category and is used in many studies for traffic flow prediction. The SVM process involves getting the optimal separating hyperplane. It can work with any number of dimensions [48]. Deshpande et al. [48], in their research, presented the potential use of SVM for traffic flow prediction. Zhou et al. [49] conducted research on traffic flow analysis based on GPS data on floating cars using a least squares (LS)–SVM method. Li et al. [50] employed SVM for bus arrival time prediction based on GPS data. Their results indicated that SVM was robust, adaptive, and able to provide good prediction accuracy.

5.2.3 Limitations of Using a Single Prediction Method

Our review reveals that several methods have been used for traffic flow prediction and modeling. These methods are classified into three categories: statistical, machine learning, and hybrid methods. The statistical methods work well only with linear traffic flow, while the machine learning methods have the ability to work with nonlinear traffic flow. Each of the above methods works well under specific conditions; when the conditions change, the performance of the predictive method is affected and the accuracy decreases.

KF exhibits high prediction accuracy, but it is a linear prediction model and it is not suitable for nonlinear traffic flow. Furthermore, it is not adaptive for dynamic traffic conditions. The KF method has limited accuracy when it deals with noisy traffic data [51].

ARIMA is a popular and widely used statistical method for traffic flow prediction. However, it has the disadvantage of being unable to capture rapid changes in traffic data. In addition, a SARIMA model requires a lot of time for estimation of parameters. ARIMA provides good performance only with static traffic conditions and does not reflect the dynamics.

Many kinds of neural networks have been used for traffic flow prediction, such as fuzzy neural models (FNMs), genetic algorithms (GAs), and multilayer perceptrons (MLPs). Large volumes of historical data and many computational resources are needed. Neutral networks are suitable for nonlinear features. Neural networks constantly demonstrate high accuracy but take more time for training and require large storage space. They are based on training using part of the historical data to find the relationship between the input and the output. Moreover, ANNs are not able to find an optimal solution for nonconvex problems.

SVMs can overcome the limitations of ANNs because they are able to map nonlinear problems in a low dimension to linear problems in a high dimension. However, they fail to give high prediction accuracy when the data contain noise.

5.2.4 Hybrid Traffic Flow Analysis and Prediction Methods

The hybrid prediction model combines the advantages of both statistical and machine learning methods. In previous studies, hybrid models have outperformed both statistical and machine learning models. However, the costs of using a hybrid model are computational complexity and large storage needs. Most of the existing hybrid models are performed using stand-alone platforms. Some existing hybrid models have been performed using a distributed platform such as Hadoop; however, the current state of work on hybrid models in distributed environments is very basic. Apache Spark has recently emerged as another big data platform with much better performance than Hadoop [52].

Wang et al. [40] conducted research on the potential use of a combination of ARIMA and SVM for traffic flow prediction. They found the characteristics of the data by using feature analysis and, on the basis of the analysis results, they used hybrid ARIMA and SVM methods. The results showed that the hybrid methods did improve the prediction accuracy.

Meng et al. [53] carried out research on the potential use of a hybrid K-nearest neighbor (K-NN) method with a balanced binary (AVL) tree for short-term traffic flow prediction to increase the accuracy of the prediction result. The results showed that the hybrid K-NN method with AVL increased the speed of the search and the accuracy outperformed both K-NN and AVL.

Xie et al. [54] proposed a novel hybrid prediction model combining the advantages of an ARIMA model and a periodical moving average (PMA) model. The model was evaluated using real-time data as well as historical data. The results showed that the forecasting performance was improved by use of the hybrid prediction model.

Li et al. [55] applied both ARIMA and a radial basis function ANN (RBF-ANN) for traffic flow prediction. The results indicated that the hybrid model had better performance than use of a single ARIMA or RBF-ANN.

5.3 Transportation Data Sources

Because of the rapid increase in the population and the numbers of vehicles, several problems have emerged in transportation systems, such as traffic congestion and traffic accidents. More recently, many types of transportation data have emerged and can be used for studying traffic status and solving transportation-related issues. Governments, city planners, and researchers have used these data for various purposes such as predicting traffic flow and identifying traffic congestion and traffic accidents. To find out the available traffic-related data, we widely review technologies used for collection and acquisition of traffic data.

On the basis of the work done by Dabiri and Heaslip [30], we can classify traffic data sources into six categories (Fig. 5.2). These are explained in Sects. 5.3.1–5.3.6.

5.3.1 Traffic Flow Sensors

Traffic flow sensors are devices for capturing the passage of vehicles over a particular road so as to capture traffic parameters. They are classified into two categories: the first category is sensors that are attached to the road pavement or road surface, such as inductive loop and magnetometers sensors; the second category is sensors that are placed above the road surface, such as infrared sensors, video image processors, and microwave radar. Both ultrasonic sensors and passive infrared sensors are widely used to collect unprocessed data, which is used by most of the existing prediction models. Prathilothamai et al. [56] proposed a system for road traffic prediction; they used traffic data collected by ultrasonic as well as passive infrared sensors, and they suggested the use of traffic video in future studies.

Fig. 5.2 Classification of traffic data sources

5.3.2 Video Image Processors

A video image processor (VIP) is a camera mounted on poles on the road pavement or traffic signal for taking images or video of passing vehicles. Microprocessors store and process these images and videos to apply computer vision algorithms for extracting traffic parameters that are used in traffic management operations.

5.3.3 Probe Vehicles and People Data

Traffic sensors and VIPs capture traffic data only in a limited area and location, which leads to data collection that is unrepresentative of the network as a whole. To overcome this limitation, individuals' vehicles equipped with GPS devices can be used for collecting representative traffic data. Also, smartphones can used to capture spatial data and can be used for tracking vehicles and people's trajectories.

Floating car data (FCD) is an important source of traffic data in smart cities and the transportation sector. It is a set of GPS entries containing information about driving status. By using FCD, traffic congestion can be identified, travel time can be computed, and traffic flow can be predicted. Castro et al. [57] proposed a method to predict future traffic conditions. They evaluated their method by using large-scale taxi GPS data obtained from around 5000 taxis in Hangzhou, China, over a period of a month (February 2010). Wang et al. [58] proposed a three-phase framework to explore the congestion correlation between road segments from three data sources: GPS trajectories of taxis, road network data, and point of interest (POI) data.

5.3.4 Social Network Data

Today, millions of people share data and communicate using social networks such as Twitter, Facebook, and Instagram. People share their images, locations, and video on social media networks. These data contain hidden knowledge and can be used in transportation systems. In social media, traffic data acquisition is performed using an application programming interface (API) by using queries to access historical and real-time information. Petalas et al. [29] proposed big data architecture for road traffic prediction using multiple sources of data. The data they used for traffic prediction were urban data and social media data. They utilized data from multiple heterogeneous sources. The results showed that the performance of the prediction model depends on the type of traffic data used.

5.3.5 Smart Card Data

Smart cards are one traffic data source, using technology for capturing transit data and passenger behavior. There are two types of smart cards: automated passenger counter (APC) cards and automated fare collection (AFC) cards. They are designed for controlling passenger movement in and out of buses and subways.

5.3.6 Environmental Data

Traffic exhibits sudden shifts due to various factors such as weather status. Meteorological data—including temperature, wind speed, and precipitation—must be taken into consideration when analyzing traffic flow. There are numerous public websites that provide metrological data, such as the [US] National Weather Service (NWS; https://www.weather.gov) and the [US] National Climatic Data Center (NCDC; https://www.ncdc.noaa.gov).

5.4 Big Data Analysis and Processing Tools

5.4.1 Hadoop

Hadoop is a powerful open-source framework (developed by Apache) used by many organizations and companies for storing, analyzing, and processing big data. Hadoop provides good availability and scalability of data [59]. Also, Hadoop is considered reliable and able to detect bugs. It is a distributed tool comprising two components: MapReduce and the Hadoop distributed file system (HDFS). Large data sets are divided into small pieces and stored in blocks of 64 MB in size; each block is called a DataNode. Those DataNodes are indexed by NameNode in HDFS [60]. MapReduce is a big data processing tool for distributed computing, which was developed by Google. MapReduce works on a divide-and-conquer principle, dividing big data problems into small problems and processing them in parallel. Hadoop also contains a data warehousing application called Hive. Hive uses structured query language (SQL) and HiveQL as query languages.

Hadoop_GIS Tool

The main components of a traditional geographic information system (GIS) are a database for storage and an analyzing model. The data are represented in a relational database or geodatabase. The data are transported to the ArcGIS environment for analysis, and the ArcGIS toolbox uses spatial analysis jobs. The traditional GIS is

single threaded, which means there is only one module to process and analyze the data stream. For this reason, the traditional GIS is not suitable for processing large data sets; the cost of processing a large data set is very high and it takes a lot of time [60]. To overcome the limitations of the traditional GIS, the Hadoop_GIS tool was proposed by Deng and Bai [60]. The Hadoop_GIS tool is a package containing a spatial framework and geoprocessing tools. The spatial framework consists of functions such as ST_Geometry. The Hadoop_GIS tool adds geometry functions and a geoprocessing toolbox for Hadoop. Hadoop_GIS is considered more efficient than the traditional GIS for processing large data sets; however, it is just a processing model with no visualization feature.

5.4.2 Apache Spark

Apache Spark is an open-source framework designed for cluster computing. It is designed to be fast and general purpose and to overcome the limitations of MapReduce. Since time is a very important factor in big data processing, Spark supports in-memory processing, which is faster than disk-based processing [61]. This feature make it faster to query big data than in a traditional disk-based engine such as Hadoop. Spark can run multiple applications in the same engine. Also, Spark has the ability to process different types of workloads that need separate systems, including queries, iterative algorithms, and streaming. This feature leads to a reduced management cost of multiple big data tools. Furthermore, the accessibility of Spark is considered very high because it provides easy application programming interfaces (APIs) in Java, Python, Scala, and SQL. Spark can be integrated with any big data tools such as Hadoop. Moreover, Spark supports additional machine learning tools such as M-Lib, a tool for graph processing (named GraphX), a tool for streaming processing (named Spark Streaming), and Spark SQL for processing structured data. All Spark components and supported tools are shown in Fig. 5.3. When one compares Spark with other big data tools such as Hadoop, Spark is faster and has greater ability to process and write data than Hadoop [61].

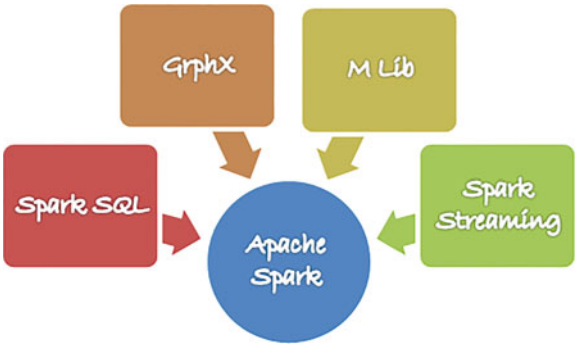


Fig. 5.3 Apache Spark components

Apache Spark features Apache Spark is considered one of the high-performance frameworks that are designed for cluster computing and real-time streaming processing [56]. It is distinguishable from all other available tools because:

- It is a general purpose framework.
- It is easy to install and configure.
- It is simple to use because it support APIs with several programming languages (such as Java, Scala, and SQL).
- It is faster than Hadoop because it supports in-memory computing.
- It supports Java and Scala, which are powerful languages for object-oriented programming.
- It has the ability to aggregate multiple data sets from different sources.
- It has the ability to join and work with other big data tools such as Hadoop.

5.5 Process of Hybrid Prediction

Hybrid methods for road traffic prediction and analysis combine both statistical and machine learning analysis. Researchers have proposed many hybrid methods with different combinations of prediction methods for several kinds of traffic data. In this section we introduce a tutorial for combining statistical analysis and machine learning analysis. Furthermore, we propose the methodology of a hybrid prediction model combining a widely used ARIMA model from the statistical category with SVM from the machine learning category.

5.5.1 Statistical Analysis

In this section, we discuss the ARIMA model as an example of a time series–forecasting method.

Autoregressive Integrated Moving Average

ARIMA was proposed by Box and Jenkins (1976) and is also called the Box–Jenkins model [55]. It is widely used in predicting stationary time series. If the series is not stationary, then it is transformed into a stationary series by differencing. The order of differencing is denoted by the parameter d. The steps for predicting traffic flow by using an ARIMA model are shown in Fig. 5.4, and an ARIMA flow chart is shown in Fig. 5.5.

1. *Data visualization*: The goal of this step is to explore any trend in the data and to decide what type of ARIMA we should use. If there is a seasonal trend in the data, we should use a seasonal ARIMA model; if there is no seasonal trend, we can use a normal ARIMA model.

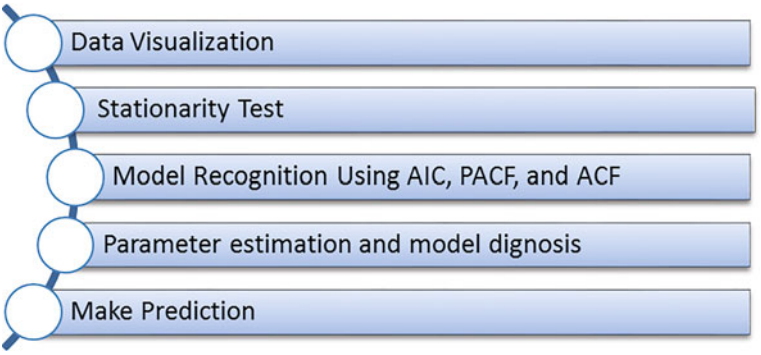
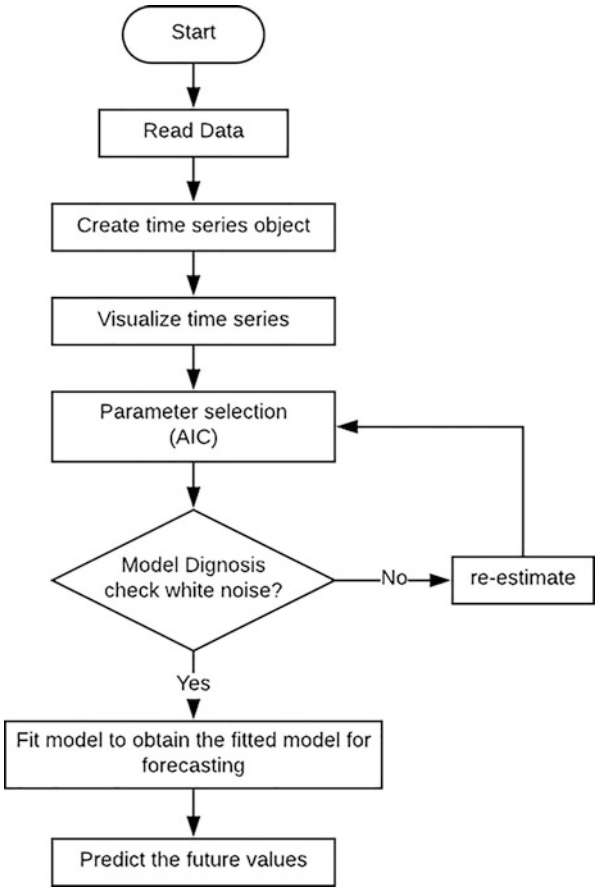


Fig. 5.4 Autoregressive integrated moving average (*ARIMA*) framework

Fig. 5.5 Autoregressive integrated moving average (*ARIMA*) flow chart



2. *Stationarity test*: Series stationary means that the mean and the variance of the series should not be a function of time. The mean of the series should not increase over time. The series is considered not stationary if there is a varying spread of the data over time. Also, the covariance of the series should not be a function of time. The covariance should be constant with time; if the spread of the data becomes closer as the time increases, then the series violates the stationary property.
3. *Model recognition*: The parameters p, d, and q are determined on the basis of the Akaike information criterion (AIC) minimum criterion, autocorrelation function (ACF), or partial autocorrelation function (PACF).
4. *Parameter estimation and model diagnosis*: The fourth step in building an ARIMA model is checking the accuracy of the model by a diagnostic test such as the Q statistic [62]. Then, we see if the chosen model and its parameters fit the data reasonably or not. If not, the parameters and the model must be re-estimated.
5. *Making the prediction*: The chosen model is used with suitable parameters to predict traffic flow.

5.5.2 Machine Leaning Analysis

Support Vector Machine

This is an advanced machine learning method and is widely used for short-term prediction such as travel time prediction and traffic flow prediction [40]. The SVM method has an algorithm called support vector regression (SVR), which is used to solve classification and regression problems [51]. SVM has the ability to predict unknown data on the basis of the given pattern. It is superior to ANN in terms of generalization and learning ability [63].

Suppose we have the training data set: $\{(x_1, y_1), \ldots (x_n, y_n)\}$. SVR can find the function that represents the relationship of x and y; also, the function gives the forecasted value of the new x. The SVR function can be represented as:

$$f(x) = w.\,\phi\,(x_i) + b \tag{5.1}$$

where w and b are the final study variables of SVR, and $\phi\,(x_i)$ is nonlinear mapping to high-dimensional space.

5.5.3 Hybrid Autoregressive Integrated Moving Average–Support Vector Machine Methodology

As we know, most real-world time series contain linear and nonlinear correlation structures. Moreover, neither ARIMA nor SVM can capture all characteristics of traffic flow patterns and provide reasonable prediction accuracy. Thus, we need to

combine ARIMA and SVM to improve the prediction accuracy. Traffic flow data contain nonlinear time series with white noise and linear time series, which can represented as:

$$Y_t = L_t + N_t \tag{5.2}$$

where L_t and N_t denote the linear and nonlinear parts, respectively. Thus, we can represent the hybrid prediction model as follows:

1. *Step 1*: Fit the ARIMA model to the linear time series, and the corresponding predicted $\acute{L}_t$ at time t is obtained.
2. *Step 2*: Compute the residual e_t from the ARIMA model as follows:

$$e_t = Y_t - \acute{L}_t \tag{5.3}$$

3. *Step 3*: Model the residual e_t by using the SVM model. Thus, the nonlinear traffic flow time series can be captured. $\acute{N}_t$ is the result of the prediction of the SVM model.
4. *Step 4*: Finally, the overall prediction value of the traffic flow time series can estimated as:

$$\acute{Y}_t = \acute{L}_t + \acute{N}_t \tag{5.4}$$

A flow diagram of the hybrid model is shown in Fig. 5.6.

5.5.4 Model Evaluation

To measure the model performance and the accuracy of the prediction results in order to compare the proposed model with other models, the following performance indexes can be used:

1. Mean absolute error (MAE):

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} \left| y(t) - y'(t) \right| \tag{5.5}$$

2. Mean square error (MSE):

$$\text{MSE} = \frac{1}{N} \sqrt{\sum_{i=1}^{N} (y(t) - y'(t))\, 2} \tag{5.6}$$

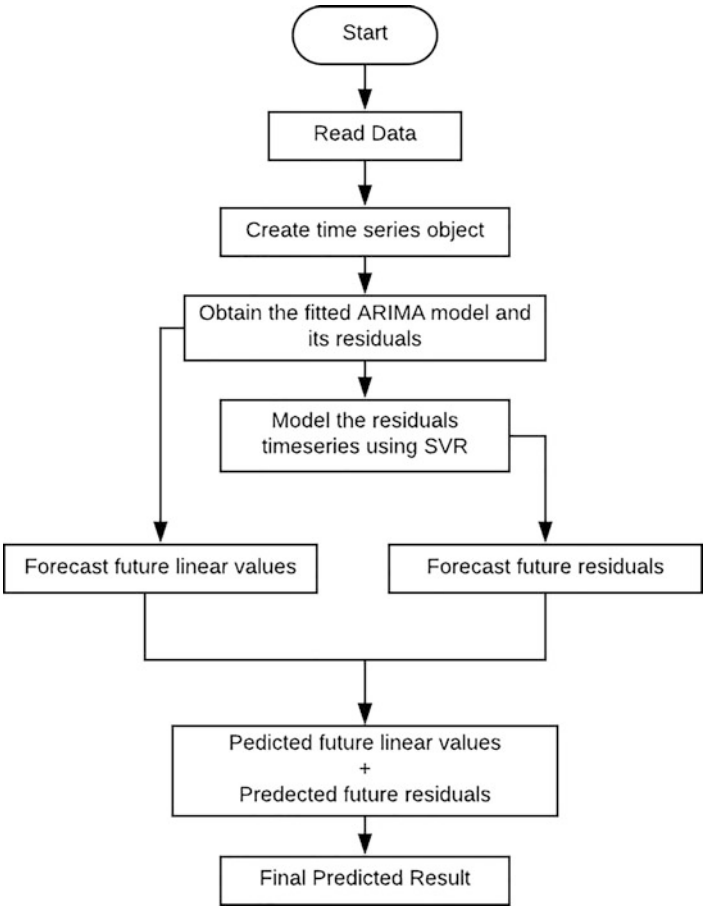


Fig. 5.6 Hybrid autoregressive integrated moving average (*ARIMA*)–support vector machine (*SVM*) model

3. Mean relative error (MRE):

$$\text{MRE} = \frac{1}{N}\left(\sum_{i=1}^{N}\frac{\left|y(t) - y'(t)\right|}{y(t)}\right) \times 100\% \tag{5.7}$$

5.6 Conclusions

In this work, we have reviewed the widely used traffic flow prediction methods along with the limitations of each method. The traffic data sources have been classified into six categories and discussed. We have also reviewed high-performance big data

analysis tools with their advantages and disadvantages. Finally, we have introduced a tutorial on the process of hybrid modeling for traffic flow prediction. The hybrid model combines both statistical and machine learning methods. In the proposed hybrid model, we combine ARIMA for linear time series and SVM for nonlinear components. The accuracy of the hybrid model can be measured using the discussed performance metrics, and this is our future work.

Acknowledgements The work carried out in this chapter is supported by the High Performance Computing (HPC) Center at the King Abdulaziz University, Jeddah.

References

1. Büscher, M., Coulton, P., Efstratiou, C., Gellersen, H., Hemment, D., Mehmood, R., Sangiorgi, D.: Intelligent mobility systems: some socio-technical challenges and opportunities. In: Communications Infrastructure. Systems and Applications in Europe, Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST 16. pp. 140–152 (2009)
2. Suma, S., Mehmood, R., Albugami, N., Katib, I., Albeshri, A.: Enabling next generation logistics and planning for smarter societies. In: Procedia Computer Science. pp. 1122–1127 (2017)
3. Suma, S., Mehmood, R., Albeshri, A.: Automatic event detection in smart cities using big data analytics. In: International Conference on Smart Cities, Infrastructure, Technologies and Applications (SCITA 2017): Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 111–122. Springer, Cham (2018)
4. Alomari, E., Mehmood, R.: Analysis of tweets in Arabic language for detection of road traffic conditions. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 98–110. Springer, Cham (2018)
5. Ayres, G., Mehmood, R., Mitchell, K., Race, N.J.P.: Localization to enhance security and services in Wi-Fi networks under privacy constraints. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 16. pp. 175–188. Springer (2009)
6. Ayres, G., Mehmood, R.: LocPriS: a security and privacy preserving location based services development framework. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), LNAI, Volume 6279, Part 4. pp. 566–575. Springer (2010)
7. Elmirghani, J.M.H., Badic, B., Li, Y., Liu, R., Mehmood, R., Wang, C., Xing, W., Garcia Zuazola, I.J., Jones, S.: IRIS: an intelligent radio-fibre telematics system. In: Proceedings of the 13th ITS World Congress, London, 8–12 October (2006)
8. Ayres, G., Mehmood, R.: On discovering road traffic information using virtual reality simulations. In: 11th International Conference on Computer Modelling and Simulation, UKSim 2009. pp. 411–416 (2009)
9. Mehmood, R.: Towards understanding intercity traffic interdependencies. In: 14th World Congress on Intelligent Transport Systems, ITS 2007. pp. 1793–1799. ITS America, Beijing (2007)
10. Mehmood, R., Nekovee, M.: Vehicular ad hoc and grid networks: discussion, design and evaluation. In: 14th World Congress on Intelligent Transport Systems, ITS 2007. pp. 1555–1562. ITS America, Beijing (2007)

11. Gillani, S., Shahzad, F., Qayyum, A., Mehmood, R.: A survey on security in vehicular ad hoc networks. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 59–74 (2013)
12. Alvi, A., Greaves, D., Mehmood, R.: Intra-vehicular verification and control: a two-pronged approach. In: 7th IEEE International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2010. pp. 401–405 (2010)
13. Schlingensiepen, J., Nemtanu, F., Mehmood, R., McCluskey, L.: Autonomic transport management systems—enabler for smart cities, personalized medicine, participation and industry Grid/Industry 4.0. In: Intelligent Transportation Systems—Problems and Perspectives, Volume 32 of the series Studies in Systems, Decision and Control. pp. 3–35. Springer (2016)
14. Schlingensiepen, J., Mehmood, R., Nemtanu, F.C.: Framework for an autonomic transport system in smart cities. Cybern. Inf. Technol. **15**, 50–62 (2015)
15. Schlingensiepen, J., Mehmood, R., Nemtanu, F.C., Niculescu, M.: Increasing sustainability of road transport in European cities and metropolitan areas by facilitating autonomic road transport systems (ARTS). In: Wellnitz, J., Subic, A., Trufin, R. (eds.) Sustainable Automotive Technologies 2013 Proceedings of the 5th International Conference ICSAT 2013, pp. 201–210. Springer, Ingolstadt (2014)
16. Alam, F., Mehmood, R., Katib, I.: D2TFRS: an object recognition method for autonomous vehicles based on RGB and spatial values of pixels. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 155–168. Springer, Cham (2018)
17. Alazawi, Z., Altowaijri, S., Mehmood, R., Abdljabar, M.B.: Intelligent disaster management system based on cloud-enabled vehicular networks. In: 2011 11th International Conference on ITS Telecommunications, ITST 2011. pp. 361–368. IEEE (2011)
18. Alazawi, Z., Abdljabar, M.B., Altowaijri, S., Vegni, A.M., Mehmood, R.: ICDMS: an intelligent cloud based disaster management system for vehicular networks. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), LNCS, Volume 7266. pp. 40–56. Springer, Vilnius (2012)
19. Alazawi, Z., Alani, O., Abdljabar, M.B., Altowaijri, S., Mehmood, R.: A smart disaster management system for future cities. In: Proceedings of the 2014 ACM International Workshop on Wireless and Mobile Technologies for Smart Cities—WiMobCity '14. pp. 1–10. ACM, New York (2014)
20. Alazawi, Z., Alani, O., Abdljabar, M.B., Mehmood, R.: An intelligent disaster management system based evacuation strategies. In: 2014 9th International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2014. pp. 673–678 (2014)
21. Alazawi, Z., Alani, O., Abdljabar, M.B., Mehmood, R.: Transportation evacuation strategies based on VANET disaster management system. Procedia Econ. Financ. **18**, 352–360 (2014)
22. Aqib, M., Mehmood, R., Albeshri, A., Alzahrani, A.: Disaster management in smart cities by forecasting traffic plan using deep learning and GPUs. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 139–154 (2018)
23. Mehmood, R., Meriton, R., Graham, G., Hennelly, P., Kumar, M.: Exploring the influence of big data on city transport operations: a Markovian approach. Int. J. Oper. Prod. Manag. **37**, 75–104 (2017)
24. Mehmood, R., Lu, J.A.: Computational Markovian analysis of large systems. J. Manuf. Technol. Manag. **22**, 804–817 (2011)
25. Mehmood, R., Graham, G.: Big data logistics: a health-care transport capacity sharing model. In: Procedia Computer Science. pp. 1107–1114 (2015)
26. Arfat, Y., Mehmood, R., Albeshri, A.: Parallel shortest path graph computations of United States road network data on Apache Spark. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 323–336. Springer, Cham (2018)

27. Arfat, Y., Aqib, M., Mehmood, R., Albeshri, A., Katib, I., Albogami, N., Alzahrani, A.: Enabling smarter societies through mobile big data fogs and clouds. Procedia Computer Science 109: 1128–1133 (2017)
28. Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. eds: Smart Societies, Infrastructure, Technologies and Applications, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST), Volume 224. Springer, Cham (2018)
29. Petalas, Y.G., Ammari, A., Georgakis, P., Nwagboso, C.: A big data architecture for traffic forecasting using multi-source information. Presented at the (2017)
30. Via, C.E., Hall, P., Tech, V., States, U., Author, C.: Transport-domain applications of widely used data sources in the smart transportation: a survey Sina Dabiri. 1–52 (2018)
31. Hadi Hosseini, S., Hadi Hosseini, S., Rahimi-kian, A.: Traffic flow prediction using MI algorithm and considering noisy and data loss conditions: an application to Minnesota traffic flow prediction. 26, 393–403 (2014)
32. Gandewar, R., Phakatkar, A.: Classification approach for big data driven traffic flow prediction using apache spark. Int. Res. J. Eng. Technol. **4**(9), 2395–2356 (2017)
33. Zhang, R., Shu, Y., Yang, Z., Cheng, P., Chen, J.: Hybrid traffic speed modeling and prediction using real-world data. In: 2015 IEEE International Congress on Big Data (BigData Congress), New York, pp. 230–237 (2015)
34. C.P.IJ. van Hinsbergen, J.W.C. van Lint, F.M.S.: Short term traffic prediction models deliverable DIIF-1a
35. Coifman, B.: Vehicle level evaluation of loop detectors and the remote traffic microwave sensor. J. Transp. Eng. **132**, 213–226 (2006)
36. Hu, J.H.J., Cao, W.C.W., Luo, J.L.J., Yu, X.Y.X.: Dynamic modeling of urban population travel behavior based on data fusion of mobile phone positioning data and FCD. 2009 17th Int. Conf. Geoinformatics. 1–5 (2009)
37. Abidin, A.F., Kolberg, M., Hussain, A.: Integrating Twitter traffic information with Kalman filter models for public transportation vehicle arrival time prediction. In: Big-Data Analytics and Cloud Computing. pp. 67–82. Springer, Cham (2015)
38. Yan, D., Zhou, J., Zhao, Y., Wu, B.: Short-term subway passenger flow prediction based on ARIMA. Presented at the December (2018)
39. Gandewar, R.R., Phakatkar, A.G., Student, M.E.: Survey on classification and prediction approaches in traffic flow. Int. J. Innov. Res. Comput. Commun. Eng. 5, (2017)
40. Wang, Y., Li, L., Xu, X.: A piecewise hybrid of ARIMA and SVMs for short-term traffic flow prediction. In: International Conference on Neural Information Processing. pp. 493–502. Springer, Cham (2017)
41. Vasantha Kumar, S., Vanajakshi, L.: Short-term traffic flow prediction using seasonal ARIMA model with limited input data. Eur. Transp. Res. Rev. 21 (2015)
42. Ma, X., Tao, Z., Wang, Y., Yu, H., Wang, Y.: Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transp. Res. Part C Emerg. Technol. **54**, 187–197 (2015)
43. Yin, C., Lin, Y., Yang, C.: A classification and predication framework for taxi-hailing based on big data. In: International Conference on Intelligent Computing ICIC 2017: Intelligent Computing Methodologies. p. pp 747–758. Springer (2017)
44. Florido, E., Castaño, O., Troncoso, A., Martínez-Álvarez, F.: Data mining for predicting traffic congestion and its application to Spanish data. In: 10th International Conference on Soft Computing Models in Industrial and Environmental Applications. pp 341–351. Springer, Cham (2015)
45. Park, D., Rilett, L.R.: Forecasting freeway link travel times with a multilayer feedforward neural network. Comput. Civ. Infrastruct. Eng. **14**, 357–367 (1999)
46. Pamuła, T.: CCIS 329—Traffic flow analysis based on the real data using neural networks. In: CCIS. pp. 364–371. Springer, Berlin (2012)

47. Ban, X., Guo, C., Li, G., Ban, X., Guo, C, Guo, C., Li, G.: Application of extreme learning machine on large scale traffic congestion prediction high-speed. In: Proceedings in Adaptation, Learning and Optimization. pp. 293–305. Springer, Cham (2016)
48. Deshpande, M., Bajaj, P.R.: Performance analysis of support vector machine for traffic flow prediction. In: 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC). pp. 126–129. IEEE (2016)
49. Zhou, X., Wang, W., Yu, L., Zhou, X., Yu, Á.L., Wang, W., Lu, W.: Traffic flow analysis and prediction based on GPS data of floating cars. In: Lecture Notes in Electrical Engineering. pp. 497–508. Springer, Berlin (2012)
50. Li, Y., Huang, C., Jiang, J.: Research of bus arrival prediction model based on GPS and SVM. In: 2018 Chinese Control And Decision Conference (CCDC). pp. 575–579. IEEE (2018)
51. Hu, W., Yan, L., Liu, K., Wang, H.: A short-term traffic flow forecasting method based on the hybrid PSO-SVR. Neural. Process. Lett. **43**, 155–172 (2016)
52. Shoro, A.G., Soomro, T.R.: Big data analysis: Ap spark perspective. Glob. J. Comput. Sci. Technol. **15**, 7–14 (2015)
53. Meng, M., Shao, C.-F., Wong, Y.-D., Wang, B.-B., Li, H.-X.: A two-stage short-term traffic flow prediction method based on AVL and AKNN techniques. J. Cent. South Univ. **22**, 779–786 (2015)
54. Xie, J., Choi, Y.-K.: Hybrid traffic prediction scheme for intelligent transportation systems based on historical and real-time data. Int. J. Distrib. Sens. Networks. **13**, 1–11 (2017)
55. Li, K.-L., Zhai, C.-J., Xu, J.-M.: Short-term traffic flow prediction using a methodology based on ARIMA and RBF-ANN. In: 2017 Chinese Automation Congress (CAC). pp. 2804–2807. IEEE (2017)
56. Prathilothamai, M., Sree Lakshmi, A.M., Viswanthan, D.: Cost effective road traffic prediction model using apache spark. Indian J. Sci. Technol. **9**, (2016). https://doi.org/10.17485/ijst/2016/v9i17/87334
57. Castro, P.S.., Zhang, D.., Li, S: Urban traffic modelling and prediction using large scale taxi GPS traces. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 57–72. Springer, Berlin (2012)
58. Wang, Y., Cao, J., Li, W., Gu, T., Shi, W.: Exploring traffic congestion correlation from multiple data sources. Pervasive Mob. Comput. **41**, 470–483 (2017)
59. Usama, M., Liu, M., Chen, M.: Job schedulers for big data processing in Hadoop environment: testing real-life schedule with benchmark programs. Digit. Commun. Networks. **3**(4), 260–273 (2017)
60. Deng, Z., Bai, Y.: Floating car data processing model based on Hadoop-GIS tools. In: 2016 Fifth International Conference on Agro-Geoinformatics (Agro-Geoinformatics). pp. 1–4. IEEE (2016)
61. Karau, H., Konwinski, A., Wendell, P., Zaharia, M.: Learning spark: lightning-fast big data analysis. (2015)
62. Zeng, D., Xu, J., Gu, J., Liu, L., Xu, G.: Short term traffic flow prediction using hybrid ARIMA and ANN models. In: 2008 Workshop on Power Electronics and Intelligent Transportation System. pp. 621–625. IEEE (2008)
63. Yang, Y., Lu, H.: Short-term traffic flow combined forecasting model based on SVM. In: 2010 International Conference on Computational and Information Sciences. pp. 262–265. IEEE (2010)

Chapter 6
Comparison of Decision Trees and Deep Learning for Object Classification in Autonomous Driving

Furqan Alam, Rashid Mehmood, and Iyad Katib

6.1 Introduction

Road transportation is among the grand global challenges affecting human lives, health, society, and economy, caused due to road accidents, traffic congestion, and other transport deficiencies. Many approaches have been proposed to address transportation challenges and develop smart transportation infrastructures, see e.g., autonomic transport systems [1–3], vehicular ad hoc network (VANETs) [4–6], vehicular systems [6, 7], disaster management [8–12], simulations [13, 14], logistics and operations research [15–19], big data [20–23], and social media based approaches [24, 25]. Autonomous vehicles are the latest among the solutions to radically address transportation challenges.

An autonomous vehicle (AV) is one that can accelerate, increase and decrease speeds, put and release brakes and steer, itself avoiding any sort of accidents. Such technology has long been part of Hollywood sci-fi quixotic vision of the future. This is due to the fact that AVs will free drivers from the boring side of driving during travel and reduce accident rates by providing breathtaking control over vehicles. In the past, many attempts have been made but subjected to the limitation of available technologies. However, in recent years with technological advancements,

F. Alam (✉) · I. Katib
Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: fmohammed0026@stu.kau.edu.sa; iakatib@kau.edu.sa

R. Mehmood
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: RMehmood@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_6

the dream of AVs comes very close to reality. Now we are able to manufacture them nevertheless they are in their testing phase. AVs have the potential to change how we look at our surroundings.

The Autonomous Driving (AD) is getting lots of attention and popularity due to its various benefits [26] and assumed to be an on-road reality soon. Most of the major industry titans which include Google, Tesla, Ford, Volvo, BMW, Microsoft, Apple, and others are making huge investments in developing technologies which will enable AD. A new forecast by Intel and Strategy Analytics research firm estimated that AVs will be a 7$ trillion market by 2050 [27]. The competition of which company will bring its AVs first on the road to common public getting so tough, resulted in various perk luring practices to get skilled engineers from the rival companies and stealing AVs technologies from the competitors [28–30]. The core of these developments revolves around the critical question, how to perceive the driving environment with higher certainties.

The key technology on which success of AVs depends is how accurately they are able to perceive the driving environment. The initial step in this quest is to recognize the static and dynamic objects around the vehicles with higher accuracies. In a driving environment, this object recognition problem is more complex due to the fact that it is multi-class problem and given the dynamic nature of the driving environment which add further complexities to it. AVs consist of several onboard and off-board sensors such as cameras, LIDAR, Radar, and GPS as illustrated in Fig. 6.1.

The aim of any object recognition system is to predict with the highest degree of certainty for the given task. The result evaluation of different classification schemes can be different in terms of classification accuracies. One classifier tends to produce better predictions for a particular class, though its overall accuracy can be lower as compared to the other. The sets of patterns of rightly classified or misclassified data instances by the distinct classifiers would not certainly coincide, thus this forms the basis to acquire better classification accuracies through decision fusion of predictions from various classifiers.

Supervised machine learning algorithms learn using a training dataset which contains independent variables and their response variables. They keep on learning until the minimum possible classification error achieved. In our previous work [31],

Fig. 6.1 General view of the autonomous driving environment

we developed a methodology to integrate supervised learning and decision fusion to enhance object classification accuracy in a driving environment, which could enable an auto-pilot to take better driving decisions. This problem equates to pixel classification. Our study revealed that the C5.0 decision tree classifier performs similar to deep learning. This paper extends and investigates the topic further and compares in depth the performance of deep learning and C5.0 decision tree classifier for object classification accuracy in driving environments. The terms "deep learning" and "feed-forward deep neural networks classifier" are interchangeably used in this paper.

The contributions of this paper are:

- We manually label 100 images from a subset of KITTI road dataset [32] by using free-form selection (polygon) rather than a box or rectangular selection. This means, highly accurate pixel labeling is achieved by carefully selecting only the area of interest to enhance the training of the algorithm.
- We compare the C5.0 decision tree classifier with feed-forward deep neural networks classifier and critically analyze both classifiers based on various performance evaluation parameters including accuracy and prediction speed.
- We investigate in detail whether there is a competitive alternative to deep learning classifiers for pixel classification problems.

The paper is divided into eight sections. Section 6.2, contains a literature review and in Sect. 6.3 we explained the dataset and data preparation for this work. Further in Sect. 6.4, the classifiers are discussed which are used in this work, whereas in Sect. 6.5, the proposed method has been explained in detail. We represent the results of the proposed method in Sect. 6.6 and the comparison results in Sect. 6.7. Finally, conclusions are drawn in Sect. 6.8.

6.2 Literature Review

Machine learning is a mighty artificial intelligence (AI) tool which helps us to understand the complicated world around us by learning. Nowadays machine learning is applied in almost every field such as biomedical, education, business, security, robotics, weather forecasting, networking, and much more [33–37]. Machine learning which eventually uses to develop AI for autonomous vehicles (AVs), ranges from infotainment systems to advanced driver assistance systems (ADAS) and further to complete self-driving auto-pilots. With machine learning, AI systems continuously learn from experience by their ability to foresee and identify the happenings in their surroundings, which is promising to be highly constructive when integrated into a software architecture of AVs. Search engine giant Google and Tesla have been doing considerable research and development for developing the AI capabilities for their autonomous cars, albeit in a more vocal manner than their counterparts. Perceiving driving environment is the key problem for facilitating safe and smooth autonomous driving. The problem starts with recognizing static

objects (road, speed breakers, traffic light, buildings) and dynamic objects (cars, cycles, trucks) around AVs. All the different objects must be classified by an object recognition system which is a multi-class problem for AVs and has been well studied in [38–40].

Identifying, tracking, and avoiding human beings is a pivotal capability of AVs. Pedestrian recognition must guarantee the safety of humans walking on footpaths and crossing the roads while auto-pilots are driving AVs which is studied in [41–43]. At Google, research scientist, Anelia Angelova introduced a novel pedestrian detection system that only requires video images [44]. Similarly to [44] in [45], the deep learning based video-only pedestrian detection system is presented which is under development at the University of California, San Diego. Works like [44, 45] could make human detection systems for AVs to pinpoint humans using low-cost sensors like cameras alone without using expensive Lidar units which can reduce the cost of AVs considerably with high reliability. The developments in [44, 45] also support the arguments of Tesla CEO Elon Musk against using expensive Lidar technology for self-driving cars. A realistic situation can arise when AVs will have a sudden encounter with a pedestrian, to save lives and avoid collision with a pedestrian is a crucial and complex problem. In paper [46], the author studies the problem of detecting sudden pedestrian encounters to aid drivers to avert any sort of accident. Road detection is a crucial problem for AVs as it decides how much space is available for driving and turning to ensure safe and smooth driving. In recent years, a lot of development has been seen in this area [47–49]. For this purpose in [50], the authors proposed a road detection technique using SVM which automatically updates the training data to minimize classification error. Similarly, in [51], linear SVM is used for Segment-Based Fractal Texture Analysis (SFTA) and compared with the multi-layer convolutional neural network (CNN). Both linear SVM and CNN produced very high classification accuracies. However, CNN showed slightly better specificity.

Another way to perceive the driving environment is to combine multiple decisions or multiple sensor data for deducing the driving environment. This can also be defined as data fusion which is well studied from various perspectives in one of the latest and comprehensive surveys [52]. The paper review mathematical methods for data fusion, specific sensor environments. Further authors discussed the emerging trends which would be benefited from data fusion [52]. For example, combining GPS and camera images to predict safe driving distance to another vehicle on the road. Combining the multiple inputs or features into a single output is a complex problem but the outcome tends to show more certainty than single sensor data analytics as achieved in proceeding literature. For example in [53] authors fuse cameras images and LIDAR for deducing driving environment by labeling segments of images, whereas in [54] object grid maps are created by combining camera images and laser. In literature such as [41, 42], the single feature set is used to identify humans. Solving the same problem, though using multi-sensor data, a smoothing-based depth up-sampling method for human detection is proposed in [55] which fuses camera images and LIDAR data. Furthermore in [56], authors use knowledge of object classes to recognize humans, car obstacles, and bicyclists. A

multi-layer perceptron (MLP) classifier is used in [57] to recognize, interpret, and track autonomous moving objects. Blend of stereo vision, LIDAR, and stereo vision data is used and supplied to MLP in [57] as input. Hane et al. use images from cameras with wheel odometry for drawing out static obstacles [58], whereas in [59] Dempster Shafer theory of evidence is used to integrate sensors data to classify the obstacles.

Combining the results of multiple classifiers tends to produce better results, this is a well-proven concept. This sort of combination is known as Decision Fusion (DF). However, it is important to select a combination of right classifiers in order to take benefits from DF. In one of such work [60], authors critically examine the use of the ρ-correlation as a way to quantify the classifier diversity for selecting classifiers for fusion. DF methods are used successfully for image classification problems. A scheme to aggregate the results of different classifiers is proposed in [61]. Situations, where the classifiers disagree with each other [34], are solved by computing the pointwise accuracy and finding the global reliability [62]. Traditional methods for hyperspectral image classification typically use raw spectral signatures without considering spatial characteristics. In work [63], a classification algorithm based on Gabor features and decision fusion is proposed. First, the adjacent and high correlated spectral bands are intelligently grouped by coefficient correlation matrix. Following that, Gabor features in each group are extracted in PCA-projected subspaces to quantify local orientation and scale characteristics. Afterward, locality preserving non-negative matrix factorization is incorporated to reduce the dimensionalities of these feature subspaces. Finally, the classification results from Gaussian-mixture-model classifiers are merged by a decision fusion rule. Experimental results show that the proposed algorithms substantially outperform the traditional and state-of-the-art methods. Majority of AVs researches are based on binary classification problems and less attention has been given to challenging multi-class problems.

In this paper, we considered the binary classification problem of road detection. One of the primary tasks of AVs is to drive within the drivable area available to them. Driving surface identification is an important and critical task in the overall success of AVs in the future [64]. Several methods in the past have been proposed to solve this problem [65–67]. To the best of our knowledge, no literature exists with respect to autonomous driving that takes leverage from connecting vehicles paradigm and information fusion for better driving scene understanding.

Road detection for the AD is an important problem. In recent years, a lot of development has been seen particularly focused on road/lane detection methods [47–49]. The paper [47] proposed an algorithm for AVs for road shape identification with the help of LIDAR by identifying geometric features. Data is fused from multiple LIDAR for identifying geometric features like berms and curbs as well as obstacles. Road shape is represented using Taylor series expansion used in [68] as:

$$y(x) = y_0 + \tan(\varnothing)\, x + C_0 \frac{x^2}{2} + C_1 \frac{x^3}{6} \tag{6.1}$$

where lateral offset between road and AV, curvature and curvature rate of the road, and angle of the road relative to the AV are represented by (y_0, C_0, C_1, ϕ). Each cell in obstacle map $M_{ij} \in \{unseen, empty, small, medium, lethal\}$. For road tracking, several observation models are possible. For [47], authors selected the model based on exponential density as:

$$p\,(x|y) \propto \mathrm{e}^{-C(x,z\,)} \tag{6.2}$$

where $C(x, z)$ is the weighted sum of possible objectives. The model provides smooth turning. Using the algorithm, AV drives through roadways successfully without the need of different parameter settings for different driving environments. Urban scenes may present additional challenges such as intersections, multi-lane scenarios, or clutter due to heavy traffic.

In [48], the authors present an integrative lane detection approach which can work in real time with a significant amount of adaptability based on urban and rural driving environments. It is assumed in [48] a colored forward facing camera is fixed at the center of the windshield. The lane width is given as $L = D_L + D_R$, where horizontal distance in meters from the left lane is D_L *and* D_R which is the distance in the meter from the right lane border. Images are converted to illuminant-invariants as to remove shadows. For the purpose of lane identification, the algorithm uses ridge which is low-level image feature that measure of crease-ness and road geometry estimation is done by Random Sample Consensus (RANSAC) algorithm, which is fed up with detected ridges. Simulations are done on both pre-scan and KITTI datasets [69]. Quantitative evaluation has been done by using pixel-wise measures, which are precision (P), recall (R), and effectiveness (F) given as:

$$P = \frac{\sum \mathcal{G} I_r}{\sum I_r} \tag{6.3}$$

$$R = \frac{\sum \mathcal{G} I_r}{\sum \mathcal{G}} \tag{6.4}$$

$$F = \frac{2PR}{P + R} \tag{6.5}$$

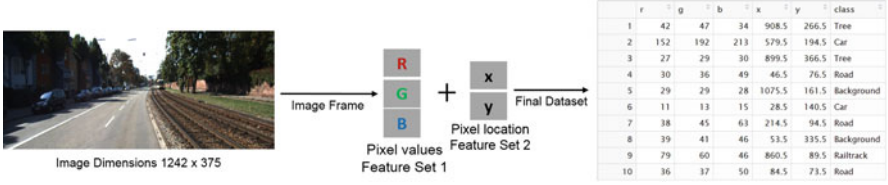
For given color images, ground-truth mask is $\mathcal{G}$ and I_r is the segmentation results. In paper [49], authors present an approach that produces reliable results exploiting a robust polyline matching technique. The proposed solution has been designed from the ground up so that only very limited hardware resources are required: just one camera is used and the processing is fast enough to be compatible with mainstream DSP units. These works focused particularly on the lane marking in urban areas where the roads infrastructure is well developed. However in underdeveloped rural areas where marking lanes is not easy due to the poor road infrastructure, few ML and deep learning (DL) based methods are proposed for road detection [70–73].

An increasing safety and reducing road accidents, thereby saving lives are one of great interest in the context of Advanced Driver Assistance Systems. Apparently, among the complex and challenging tasks of future road vehicles is road lane detection or road boundaries detection. It is based on lane detection (which includes the localization of the road, the determination of the relative position between vehicle and road, and the analysis of the vehicle's heading direction). One of the principal approaches is to detect road boundaries and lanes using the vision system on the vehicle. However, lane detection is a difficult problem because of the varying road conditions that one can encounter while driving. In paper [74], a vision-based lane detection approach capable of reaching real-time operation with robustness to lighting change and shadows is presented. The system acquires the front view using a camera mounted on the vehicle then applying few processes in order to detect the lanes. Using a pair of hyperbolas which are fitting to the edges of the lane, those lanes are extracted using a Hough transform. The proposed lane detection system can be applied to both painted and unpainted road as well as curved and straight road in different weather conditions. This approach was tested and the experimental results show that the proposed scheme was robust and fast enough for real-time requirements. Eventually, a critical overview of the methods was discussed, and their potential for future deployment was assisted [74].

6.3 Dataset and Data Preparation

We used KITTI datasets [32] in our work [31], where we have used two feature sets which are (R,G,B) values of the pixels and spatial values of each pixel in the image frames of dimension 1242 x 375 as depicted in Fig. 6.2. We create a dataset which has four attributes, namely r, g, b, x, y, class. We used Raster package [75] in R, to compute pixel values and location of each pixel in the image frame.

Further, we manually labeled the images from a subset of KITTI city dataset [32] by using free-form selection (polygon) rather than a box or rectangular selection which is further depicted in Fig. 6.3. This means highly accurate pixel labeling is achieved by carefully selecting only the area of interest to enhance the training of the machine learning algorithm. After selecting pixels of a particular object, we manually labeled every object pixel and spatial values to make the final dataset.



	r	g	b	x	y	class
1	42	47	34	908.5	266.5	Tree
2	152	192	213	579.5	194.5	Car
3	27	29	30	899.5	366.5	Tree
4	30	36	49	46.5	76.5	Road
5	29	29	28	1075.5	161.5	Background
6	11	13	15	28.5	140.5	Car
7	38	45	63	214.5	94.5	Road
8	39	41	46	53.5	335.5	Background
9	79	60	46	860.5	89.5	Railtrack
10	36	37	50	84.5	73.5	Road

Fig. 6.2 Data preparation process

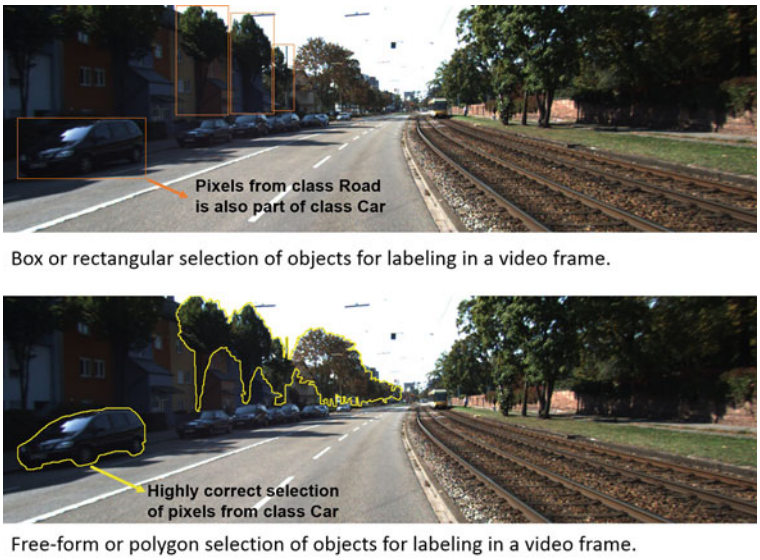


Fig. 6.3 Object labeling process. The image is taken from the KITTI dataset [32]

Our dataset in [31] contains 380,000 rows and six attributes and we divided the datasets into two parts which are training 60% and 40% testing. Further, we used the SMOTE algorithm on training data to overcome the class imbalance problem which is discussed in the proceeding section.

In this paper, we are using a bigger dataset to perform a comparative analysis between C5.0 decision tree classifier and feed-forward deep neural networks classifiers from KITTI road dataset [32]. We are using 100 video frames of the same size as used in [31]. This dataset contains 46,575,000 rows with four attributes, namely (RGB) pixels values and class labels (road, background). In this work, we are addressing the binary classification problem of predicting drivable surface (road) and non-drivable surface (background).

6.4 Algorithms

In this paper, we performed comparative analysis in which we used several supervised machine learning algorithms based on their prediction accuracy, execution time and scalability for classification and decision fusion through majority voting. In this paper, our focus is primarily on C5.0 decision tree classifier and Feed-forward deep neural networks classifiers.

6.4.1 Decision Tree

C5.0 classifier is a supervised learning algorithm which builds a decision tree using the concept of information entropy proposed by Ross Quinlan [76]. It can handle both continuous and discrete attributes. In this work we used C5.0 decision tree classifier, which is an extension of C4.5 and is also commercially sold by Ross Quinlan. The reason to use the C5.0 decision tree classifier for this work lies in the fact that it is extremely fast, several folds faster than its predecessor C4.5. It can take benefits of multi-core and multiple CPU [77]. Further, it has better memory management, which is needed because a significant amount of data processing is required particularly in RGB image classification. It can give similar or better results to C4.5 and forms significantly smaller decision trees. In Fig. 6.4, we depicted sample decision rules of C5.0 decision tree classifier for pixel classification.

C5.0 decision tree classifier is widely used in satellite image classification. Various variants of decision trees including C5.0 decision tree classifier are used in the land cover classification of satellite imagery in [78], crop classification in China's North Xinjiang area in [79] and for improving classification of land cover by pixel integration and decision trees in [80]. Further, an optimized version of the C5.0 decision tree classifier which uses Bayesian theory is proposed in [81]. C5.0 decision tree classifier is a very good choice if the dataset contains few features like in the case of the pixel, only three (RGB) values need to classify. However, C5.0 decision tree classifier learns slowly if datasets have too many features.

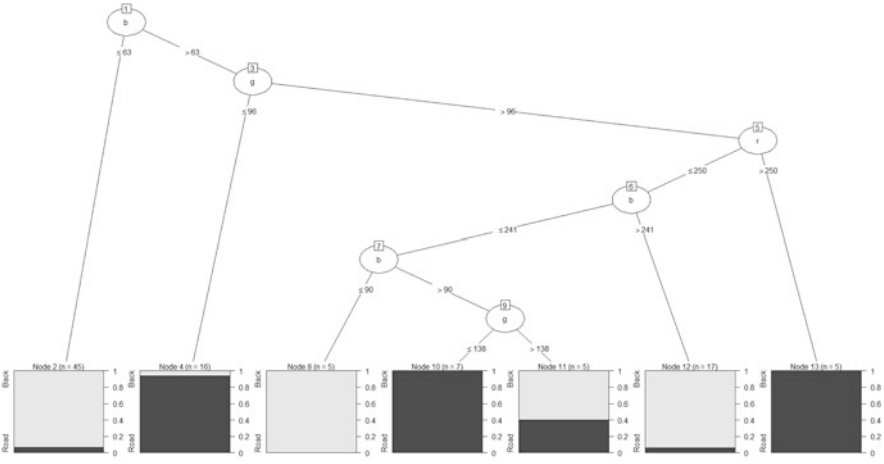


Fig. 6.4 C5.0 decision tree classifier sample rules for pixel classifications

6.4.2 Support Vector Machine

Support vector machine (SVM) is one of the most accurate classifiers and have a sound theoretical foundation. SVM constructs hyperplane or a set of hyperplanes for performing classification and regression [82]. It can compete with far more complex modern-day classifiers in terms of accuracies and it is considered one of the best classifiers which are listed among the top 10 machine learning algorithm [77]. SVM is used widely for image classification problems such as image segmentation of colored images [83, 84], image classification through remote sensing [85, 86], and face recognition [87, 88]. SVM based classifications are very accurate, particularly for binary classification problems. However, they are not fast enough if compared to deep learning algorithms.

6.4.3 Deep Learning

Deep learning (DL) mimics a neural system of humans for performing learning task. It belongs to the family of artificial neural networks. It digs deep into the data and finds out the complex relationships among data elements. Deep learning algorithms are now widely used in image recognition [89, 90], natural language processing [91–93], speech recognition [94, 95], weather forecasting [96, 97], and bioinformatics [98, 99] due to its quality of producing highly accurate predictions, though DL is computationally expensive. To develop a further understanding of various deep learning architectures, models, and their mathematical formulations in a more comprehensive manner, work such as [100–103] can be investigated further. In Fig. 6.5, we depicted classical deep learning architecture.

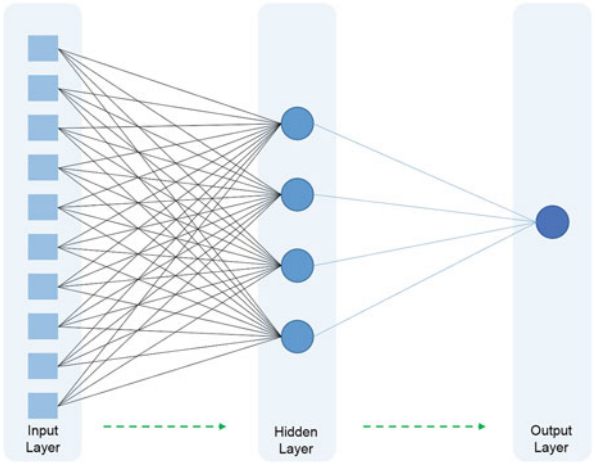


Fig. 6.5 Classical architecture of deep neural networks

6.5 Proposed Method

In this paper, our prime focus is to critically compare C5.0 decision tree classifier with feed-forward deep neural networks classifier. Whereas in work [31], our prime focus is to identify data instances which are most difficult to classify for the given supervised machine learning algorithm, prior to classifying them and to reclassify the predicted misclassified data. We divide our main method [31] into two phases. In the first phase, we carefully train our models and generate data for the training of proceeding stage because, from stage-2 onwards, machine learning algorithms need to be trained with the data specific to that stage. In the second phase, we test our whole method to predict its accuracy.

All the experimentations are performed on the R statistical machine learning platform, and H2O [104], C5.0 decision tree classifier [105], and Caret [106] libraries are used. All the simulations are carried out on the Aziz supercomputer. The Aziz supercomputer is Fujitsu made and is able to deliver peak performance of 230 teraflops. It has a total of 11,904 cores in 496 nodes. Aziz was ranked number 360 in the June 2015 Top500 competition (http://www.top500.org/); currently, it is at number 491 (November 2015).

6.5.1 Training

Formally we can define our training process as, for the given training set (X_i, Y_i), we want to generate a classifier function f to predict Y_i labels for new $X_i = (r_{i1}, g_{i2}, b_{i3}, x_{i4}, y_{i5})$. In work [31], the training process is very critical and the core of the work. It serves two purposes. Firstly, identify accurate machine learning models and secondly, generate dataset for next stage. For method depicted in Fig. 6.6, as input we used data_1 to train C5.0 decision tree classifier and to make data for training of the next stage. We predicted class labels using $data_2$.

Misclassified data instances in [31] are only 2.71% of the whole data. Training C5.0 decision tree classifier for predicting misclassified (miss) and rightly classified (hit) data labels produced results with high accuracy. However, the prediction accuracy of misclassified data instances is below 50%. This is due to the imbalance dataset problem. To counter this, we used the SMOTE algorithm [107], to generate balanced and massive data of 1.5 million rows for training C5.0 decision tree classifier for predicting miss and hit and update $data_2$ accordingly. Then we separated miss and hit data. Further miss dataset $\left(D_{miss_1}\right)$ is used to train classifiers for majority voting in [31]. Same steps are repeated to train classifiers n number of times. Class labels which are predicted at different stages are combined together in P_{final} based on row indexes of original input data.

Whereas in this paper, we are using a far bigger dataset which contains 46,575,000 rows. For training, we used 60% data that is 27,945,000 rows with class labels and for testing, we used the remaining 40% data that is 18,630,000 rows of unlabeled data.

Fig. 6.6 Training method

Input: $data_1, data_2$
Output: *Trained Models*

1. $f = \{\ Train\ M_1 = C5.0_{main}(data_1)$
2. $Predict\ P_1 \leftarrow M_1(data_2)$
3. $For(i\ to\ nrow(data_2))$
4. {
5. $If(data_2[class, i]\ !\ = P_1[i])$
6. $\{data_2[status, i] = "miss"\}$
7. }
8. $MJ_1 \leftarrow [C5.0_1(D_{miss_1}), SVM_1(D_{miss_1}), DL_1(D_{miss_1})]$
9. # Accuracy Table of each classifier and Majority Vote
10. $T_1 \leftarrow [P(MJ_1), P(C5.0_1), P(SVM_1), P(DL_1)]$
11. # Function to select classifier or Majority vote based #on highest classification accuracy and named as M_2
12. $M_2 \leftarrow maxAcc(T_1)$
13. $Predict\ P_2 \leftarrow M_2(D_{miss_1})$
14. $For(i\ to\ nrow(D_{miss_1}))$
15. {
16. $If(D_{miss_1}[class, i]\ !\ = P_2[i])$
17. $\{D_{miss_2} = D_{miss_1}[i]\}$
18. }
20. Repeat from line 8 to 16, n times, incrementally and save each trained classifier so to use them during testing
21. $Predict\ P_{final} \leftarrow MergeRowsIndexwise(P_1, P_2, P_3, \ldots . P_n)\}$

6.5.2 Testing

The testing process is explained in Fig. 6.7, which is self-explanatory in nature. We used the classier function f obtained from the training process, to predict Y_i label for new $X_i = (r_{i1}, g_{i2}, b_{i3}, x_{i4}, y_{i5})$. All trained classifiers are used in the testing phase. In Fig. 6.7, from stage-2 all the steps are repeated n number of times. In this work, we used $n = 2$; however, it can be more but will reduce the prediction speed.

6.5.3 Comparison

Our previous work [31] strongly pointed out that the C5.0 decision tree classifier performed very close to feed-forward deep neural networks classifier. Therefore to have more convincing evidence for this, in this paper, we compared the above-mentioned classifiers. While testing D2TFRS [31], we used a small dataset as explained in Sect. 6.3. However, for this work, we used a bigger dataset which contains more than 40.5 million rows. To speed up C5.0 decision tree classifier prediction, we process it in parallel using a thick node of Aziz supercomputer which has 24 cores and 256 GB memory [108]. First, we trained the C5.0 decision tree classifier, then the dataset is broken down into 24 sub-datasets. Each core of Aziz process trained C5.0 decision tree classifier to predict each sub-dataset and 24 cores predict together in parallel. This is done to match-up the prediction time with feed-forward deep neural networks. In Fig. 6.8, we depicted the parallel execution model for the C5.0 decision tree classifier.

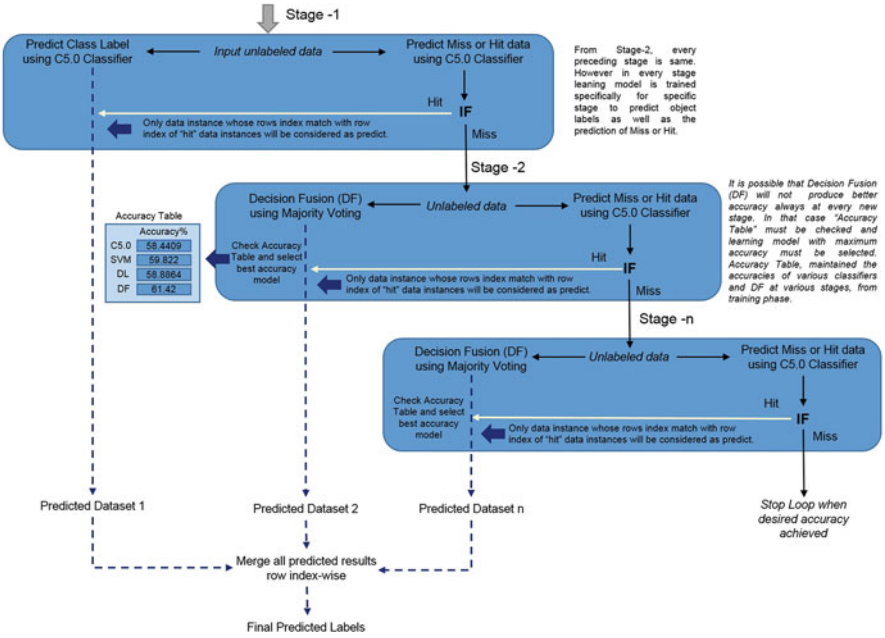


Fig. 6.7 Block diagram of proposed D2TFRS

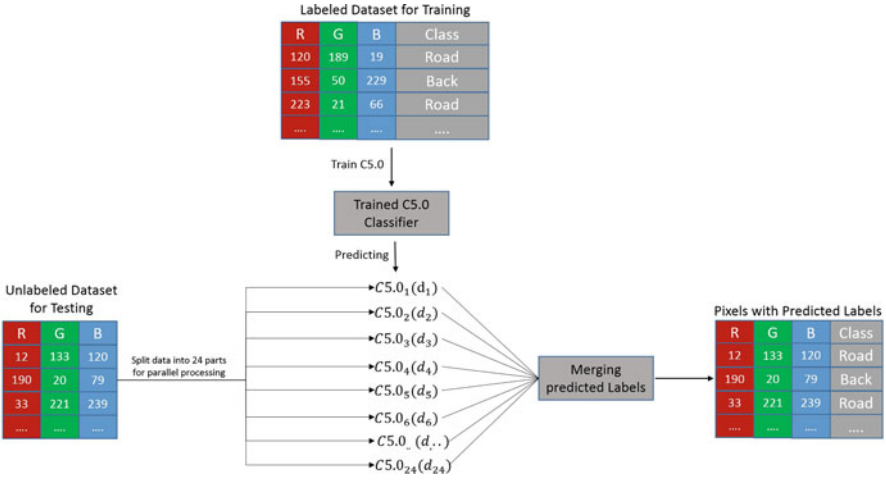


Fig. 6.8 Block diagram of C5.0 decision tree classifier execution in parallel

We constructed feed-forward deep neural networks classifier, to find the best parameters for obtaining best results we performed hyper-parameter tuning through grid search using Aziz [108]. We achieved best accuracy of 93.89269% for feed-forward deep neural networks model with epochs (1000), hidden (64,64,64), activation (rectifier), L1 (0), L2 (0), rate (0.005), adaptive_rate (False), rate_annealing

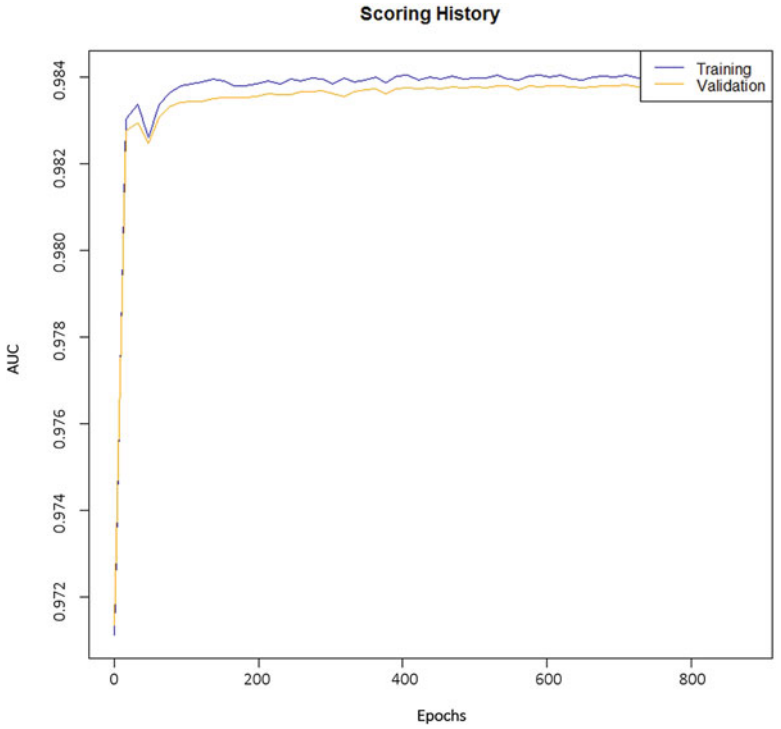


Fig. 6.9 AUC curve training vs validation for deep learning

(1e-06), input_dropout_ratio (0.1), and stopping_metric (AUC). The area under curve (AUC) plot training vs validation of feed-forward deep neural networks classifier is given in Fig. 6.9.

6.6 Results and Analysis

To evaluate our results, we compared D2TFRS method to C5.0 decision tree and AdaBoost classifiers. We used confusion matrix, sensitivity, and specificity as the benchmarks for results evaluation.

6.6.1 Confusion Matrix

A confusion matrix (CM) is a table which shows actual versus predicted data labels. The sum of diagonal (SoD) of CM represents the correctly classified data label, thus it can be used to compute classifier accuracy too which can be given as:

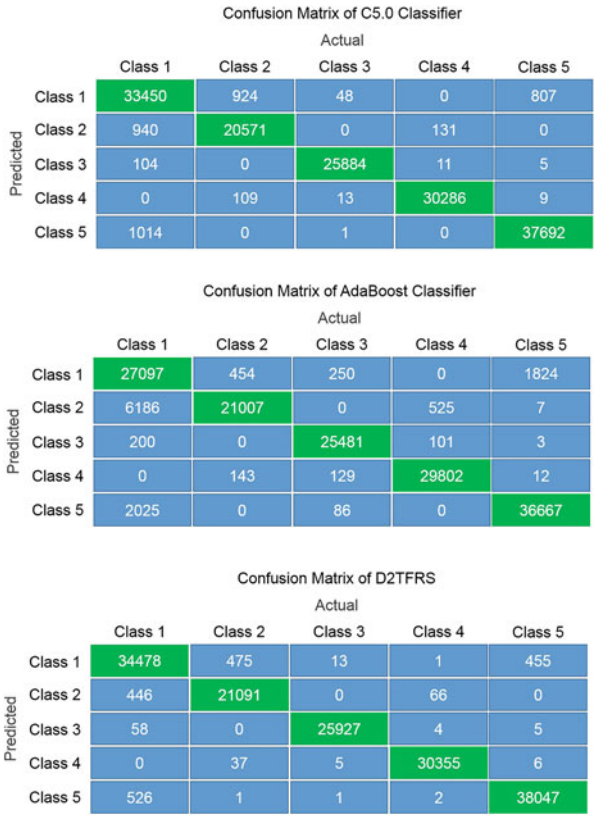


Fig. 6.10 Confusion matrix of C5.0, AdaBoost, and D2TFRS

$$\text{Accuracy}\% = (\text{SoD}/\text{Sum of all cells of CM}) \times 100 \tag{6.6}$$

In Fig. 6.10, we visualize the CM of the D2TFRS method, C5.0 decision tree, and AdaBoost classifiers. SoD which is the green color cells in Fig. 6.6, for each classifier represent rightly classified data labels. D2TFRS outperformed the AdaBoost classifier by getting 6.48% better classification accuracy. D2TFRS performed better than C5.0 decision tree classifier which produces a classification accuracy of 97.29%, which is 1.33% less than the classification accuracy of D2TFRS.

6.6.2 Sensitivity and Specificity

Sensitivity can be defined as the proportion of actual class labels which are correctly predicted by the classifier, whereas specificity is the ability of the classifier to identify negative results. Important terms used to calculate sensitivity and specificity

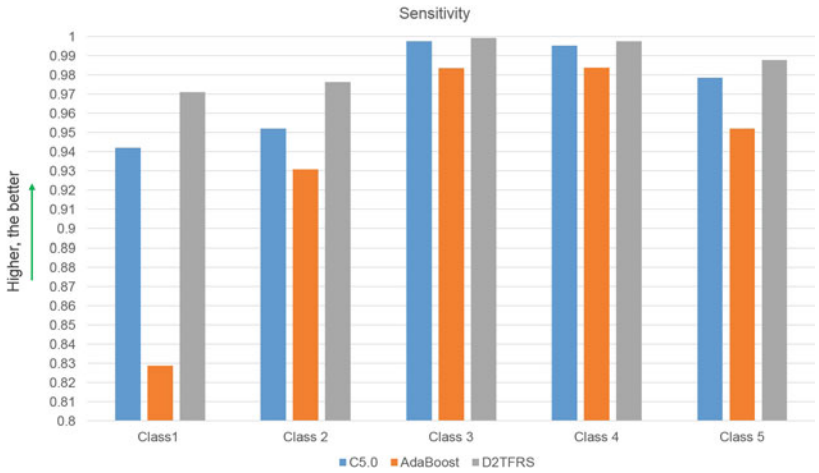


Fig. 6.11 Sensitivities measurement of C5.0, AdaBoost, and D2TFRS

are a number of true positive (TP), number of true negatives (TN), number of false positive (FP), and number of false negatives (FN), respectively.

Mathematically, these can be expressed as:

$$\text{Sensitivity} = \text{TP}/\left(\text{TP} + \text{FN}\right) \tag{6.7}$$

$$\text{Specificity} = \text{TN}/\left(\text{TN} + \text{FP}\right) \tag{6.8}$$

In terms of sensitivity and specificity, D2TFRS performed better than C5.0 decision tree and AdaBoost classifiers for all classes as depicted in Figs. 6.11 and 6.12. AdaBoost performed worst among the three, whereas C5.0 decision tree classifier performed better than AdaBoost but lacks slightly behind proposed D2TFRS. Further, a graphical comparison of sensitivities and specificities are given in Figs. 6.11 and 6.12.

6.6.3 Kappa and Speed

Kappa (κ) is an index that considers an observed agreement with respect to a baseline agreement [109]. κ is a statistical benchmark to measure classification. There are no universal acceptability criteria on how to interpret κ. However first of its kind guidelines are given by Landis and Koch. Value of κ nearer to 1 means substantial or almost perfect agreement, whereas the value of κ farther from 1 means no agreement or slight agreement. For more detail, characterization of κ can be found in [110].

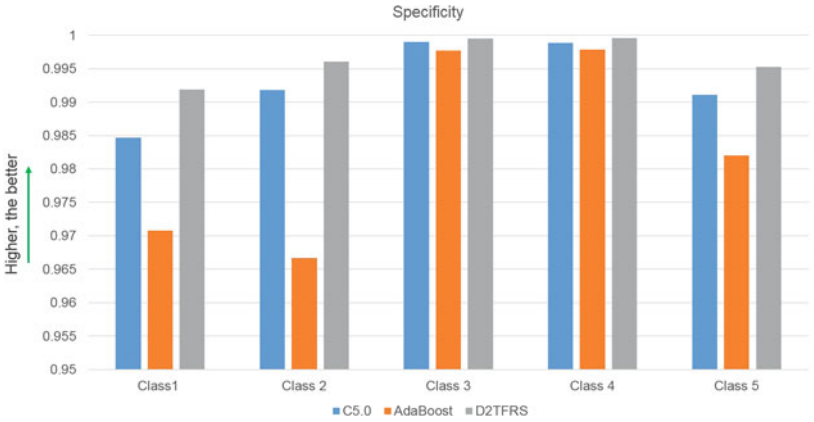


Fig. 6.12 Specificity measurement of C5.0, AdaBoost, and D2TFRS

Table 6.1 Hyper-parameter Tuning

Parameters	Values
Epochs	100, 400, 800, 1000, 2000
Hidden	(32,32), (64,64), (128,128), (256,256), (32,32,32), (64, 64, 64), (128, 128, 128), (256, 256, 256)
L1	0, 1e-3, 1e-5
L2	0, 1e-3, 1e-5
Activation	Tanh, TanhWithDropout, Maxout, Rectifier, RectifierWithDropout, MaxoutWithDropout
Input dropout ratio	0, 0.1, 0.05
Rate	0.01, 0.02, 0.005
Rate annealing	1e-8, 1e-7, 1e-6
Adaptive rate	True, False
Stopping metric	AUC, RMSE

Mathematically, κ can be expressed as:

$$\kappa = (p_o - p_e) / (1 - p_e) \tag{6.9}$$

where an observed agreement is given as p_o and expected agreement is given as p_e. The value of κ is always ≤ 1. Values of κ are given in Table 6.1. Proposed method, D2TFRS, has an almost perfect agreement which is nearest to 1 as compared to C5.0 decision tree and AdaBoost classifiers.

In terms of speed, C5.0 decision tree classifier took 5.11 s which is almost five times faster than D2TFRS which took 24.09 s and AdaBoost took 125 s with 20 iterations for which boosting is run. We strongly believe parallelization can increase the speed of D2TFRS by several magnitudes, as in this work D2TFRS implemented sequentially not in parallel. Further details of accuracy, Kappa, and speeds can be found in Table 6.2.

Table 6.2 Classification statistics

	Accuracy (%)	Kappa	Speed (s)
C5.0	97.29	0.9658	5.11
AdaBoost	92.14	0.9012	125
D2TFRS	98.62	0.9825	24.9

6.7 A Detailed Comparison of C5.0 and Deep Learning Results

For comparative analysis, we used confusion matrix, sensitivity, specificity, kappa, prediction accuracy, and time taken to predict as performance evaluation parameters which is explained in Sect. 6.6 in detail. In terms of prediction accuracy, both classifiers performed very close. The prediction accuracy of the C5.0 decision tree classifier is 93.9367%, whereas feed-forward deep neural networks are 93.8926%. In terms of sensitivity, C5.0 decision tree classifier performed slightly better than feed-forward deep neural networks classifier; however, in terms of specificity feed-forward deep neural networks classifier performed vice versa.

We implemented the C5.0 decision tree classifier in parallel so to match-up the prediction speed with feed-forward deep neural networks classifier. Despite parallel implementation, C5.0 decision tree classifier took 69 seconds whereas feed-forward deep neural networks classifier took 105 seconds. This clearly shows that feed-forward deep neural networks classifier outperformed C5.0. Further, Figs. 6.13 and 6.14 can be referred for detailed results in this respect.

6.8 Conclusion

Autonomous vehicles (AVs) are not anymore a future imagination of vehicles which can drive itself without any sort of human assistance. AVs are nowhere though they are in the final phase of testing before public use. In this paper, we compared the C5.0 decision tree classifier with feed-forward deep neural networks classifier. To maximize the speed of C5.0 decision tree classifier by several magnitudes, we implemented it in parallel. We are able to minimize prediction speed of C5.0 decision tree classifier considerably using a combination of parallel programming and high performance computing through Aziz supercomputer.

This comparative analysis is specifically for pixel-based classification problems. In this paper, we focused on AVs. We clearly see that the C5.0 decision tree classifier performs at par with feed-forward deep neural networks classifier in terms of classification accuracy. However, in terms of prediction speed, feed-forward deep neural networks classifier outperforms C5.0 decision tree classifier even after parallelization. Our work in this paper shows that for pixel classification problems C5.0 decision tree classifier can be a good choice and an alternative to resource exhausting deep learning algorithms if prediction speed does not matter much. However, for near real-time and real-time predictions, deep learning algorithms hold the edge.

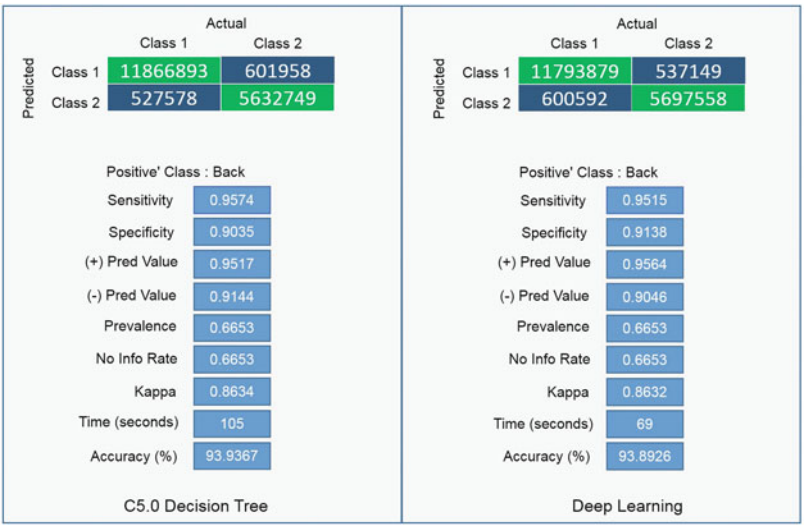


Fig. 6.13 Simulation results of the comparative analysis

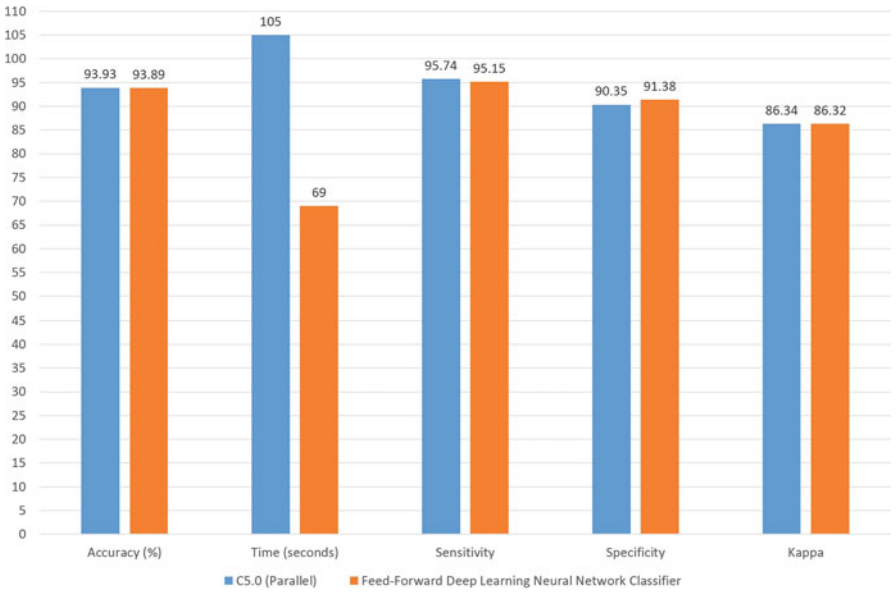


Fig. 6.14 Graphical comparisons of results

Acknowledgments The authors acknowledge with thanks the technical and financial support from the Deanship of Scientific Research (DSR) at the King Abdulaziz University (KAU), Jeddah, Saudi Arabia, under the grant number G-673-793-38. The work carried out in this paper is supported by the HPC Center at KAU.

References

1. Schlingensiepen, J., Mehmood, R., Nemtanu, F.C., Niculescu, M.: Increasing Sustainability of Road Transport in European Cities and Metropolitan Areas by Facilitating Autonomic Road Transport Systems (ARTS). In: Wellnitz, J., Subic, A., Trufin, R. (eds.) Sustainable Automotive Technologies 2013 Proceedings of the 5th International Conference ICSAT 2013, pp. 201–210. Springer International Publishing, Ingolstadt, Germany (2014)
2. Schlingensiepen, J., Nemtanu, F.: Autonomic Transport Management Systems—Enabler for Smart Cities, Personalized Medicine, Participation and Industry Grid/Industry 4.0. In: Sladkowski, A., Pamula, W. (eds.) Intelligent Transportation Systems – Problems and Perspectives, pp. 3–35. Springer International Publishing, London (2016)
3. Schlingensiepen, J., Mehmood, R., Nemtanu, F.C.: Framework for an autonomic transport system in smart cities. Cybern. Inf. Technol. **15**, 50–62 (2015)
4. Mehmood, R., Nekovee, M.: Vehicular AD HOC and grid networks: Discussion, design and evaluation. In: 14th World Congress on Intelligent Transport Systems, ITS 2007. pp. 1555–1562 (2007)
5. Gillani, S., Shahzad, F., Qayyum, A., Mehmood, R.: A Survey on Security in Vehicular Ad Hoc Networks. (2013)
6. Alvi, A., Nabi, Z., Greaves, D.J., Mehmood, R.: Intra-vehicular verification and control: a two-pronged approach. Int. J. Veh. Inf. Commun. Syst. **2**, 248–268 (2011)
7. Nabi, Z., Alvi, A., Mehmood, R.: Towards standardization of in-car sensors. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). LNCS. **6596**, 216–223 (2011)
8. Alazawi, Z., Altowaijri, S., Mehmood, R., Abdljabar, M.B.: Intelligent disaster management system based on cloud-enabled vehicular networks. In: Vinel, A., Berbineau, M., Manohara, P.M.M., Koucheryavy, Y., Gusikhin, O., Prokhorov, D., Rodrigues, J., Zhang, Y. (eds.) 2011 11th International Conference on ITS Telecommunications, ITST 2011, pp. 361–368. IEEE, St. Petersburg, Russia (2011)
9. Alazawi, Z., Abdljabar, M.B., Altowaijri, S., Vegni, A.M., Mehmood, R.: ICDMS: An intelligent cloud based disaster management system for vehicular networks. (2012)
10. Alazawi, Z., Alani, O., Abdljabar, M.B., Altowaijri, S., Mehmood, R.: A Smart Disaster Management System for Future Cities. WiMobCity'14. Int. Work. Wirel. Mob. Technol. Smart Cities. 1–10 (2014)
11. Aqib, M., Mehmood, R., Albeshri, A., Alzahrani, A.: Disaster management in smart cities by forecasting traffic plan using deep learning and GPUs. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST. pp. 139–154. Springer, Cham (2018)
12. Alazawi, Z., Alani, O., Abdljabar, M.B., Mehmood, R.: An intelligent disaster management system based evacuation strategies. In: 2014 9th International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2014. pp. 673–678 (2014)
13. Ayres, G., Mehmood, R.: On discovering road traffic information using virtual reality simulations. In: 11th International Conference on Computer Modelling and Simulation, UKSim 2009. pp. 411–416 (2009)
14. Mehmood, R.: Towards understanding intercity traffic interdependencies. In: 14th World Congress on Intelligent Transport Systems, ITS 2007. pp. 1793–1799 (2007)
15. Mehmood, R., Lu, J.A.: Computational Markovian analysis of large systems. J. Manuf. Technol. Manag. **22**, 804–817 (2011)
16. Büscher, M., Coulton, P., Efstratiou, C., Gellersen, H., Hemment, D., Mehmood, R., Sangiorgi, D.: Intelligent mobility systems: Some socio-technical challenges and opportunities. (2009)
17. Mehmood, R., Meriton, R., Graham, G., Hennelly, P., Kumar, M.: Exploring the influence of big data on city transport operations: a Markovian approach. Int. J. Oper. Prod. Manag. **37**, 75–104 (2017)

18. Mehmood, R., Graham, G.: Big Data Logistics: A health-care Transport Capacity Sharing Model. Procedia Comput. Sci. **64**, 1107–1114 (2015)
19. Graham, G., Mehmood, R., Coles, E.: Exploring future cityscapes through urban logistics prototyping: a technical viewpoint. Supply Chain Manag. **20**, 341–352 (2015)
20. Arfat, Y., Mehmood, R., Albeshri, A.: Parallel shortest path graph computations of United States road network data on apache spark. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST. pp. 323–336. Springer, Cham (2018)
21. Arfat, Y., Aqib, M., Mehmood, R., Albeshri, A., Katib, I., Albogami, N., Alzahrani, A.: Enabling Smarter Societies through Mobile Big Data Fogs and Clouds. In: Procedia Computer Science (2017)
22. Suma, S., Mehmood, R., Albugami, N., Katib, I., Albeshri, A.: Enabling Next Generation Logistics and Planning for Smarter Societies. In: Procedia Computer Science (2017)
23. Usman, S., Mehmood, R., Katib, I.: Big Data and HPC Convergence: The Cutting Edge and Outlook. Presented at the November 27 (2018)
24. Suma, S., Mehmood, R., Albeshri, A.: Automatic Event Detection in Smart Cities Using Big Data Analytics. In: International Conference on Smart Cities, Infrastructure, Technologies and Applications SCITA 2017: Smart Societies, Infrastructure, Technologies and Applications. pp. 111–122. Springer, Cham (2018)
25. Alomari, E., Mehmood, R.: Analysis of tweets in Arabic language for detection of road traffic conditions. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST. pp. 98–110. Springer, Cham (2018)
26. Litman, T.: Autonomous Vehicle Implementation Predictions Implications for Transport Planning. Transp. Res. Board Annu. Meet. **42**, 36–42 (2015)
27. Morris, D.Z.: Driverless Cars Will Be Part of a $7 Trillion Market by 2050, http://fortune.com/2017/06/03/autonomous-vehicles-market/, (2017)
28. McGoogan, C.: Uber fires driverless car boss accused of stealing Google's trade secrets, http://www.telegraph.co.uk/technology/2017/05/31/uber-fires-driverless-car-boss-failing-assist-google-lawsuit/, (2017)
29. Kharpal, A.: Apple has reportedly hired ex-NASA and Tesla staffers to boost its self-driving car effort, https://www.cnbc.com/2017/04/25/apple-driverless-cars-hires-nasa-tesla.html
30. Feris, R.: Tesla sues former Autopilot director for allegedly stealing secrets, poaching coworkers, https://www.cnbc.com/2017/01/26/tesla-sues-former-exec-for-allegedly-stealing-secrets-poaching-workers.html
31. Alam, F., Mehmood, R., Katib, I.: D2TFRS: An Object Recognition method for Autonomous Vehicles based on RGB and Spatial Values of Pixels. Mehmood R., Bhaduri B., Katib I., Chlamtac I. Smart Soc. Infrastructure, Technol. Appl. SCITA 2017. Lect. Notes Inst. Comput. Sci. Soc. Informatics Telecommun. Eng. Springer. 224, 155–168 (2017)
32. Geiger, A., Lenz, P.: Vision meets Robotics: The KITTI Dataset. Int. J. Robot. Res. (2013)
33. Mehmood, R., Alam, F., Albogami, N.N., Katib, I., Albeshri, A., Altowaijri, S.: UTiLearn: A Personalised Ubiquitous Teaching and Learning System for Smart Societies. IEEE Access. **3536**, 1–22 (2017)
34. Alam, F., Mehmood, R., Katib, I., Albeshri, A.: Analysis of Eight Data Mining Algorithms for Smarter Internet of Things (IoT). Int. Work. Data Min. IoT Syst. (DaMIS 2016). 98, 437–442 (2016)
35. Alam, F., Thayananthan, V., Katib, I.: Analysis of Round-robin Load-balancing Algorithm with Adaptive and Predictive Approaches. 11th Int. Conf. Control. (2016)
36. Raissi, M.: Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations. Cornell Univ, Libr (2018)
37. Alam, F., Mehmood, R.: Tutorial: Data Analytics for Internet of Things. High Perform. Comput. Conf, Saudi Arab (2018)
38. Andriluka, M., Roth, S., Schiele, B.: People-Tracking-by-Detection and People-Detection-by-Tracking. IEEE Conf. Comput. Vis. Pattern Recognition. **2008**, (2008)

39. Petrovskaya, A., Thrun, S.: Model based vehicle detection and tracking for autonomous urban driving. Auton. Robots. 123–139 (2009)
40. Wu, B.O., Nevatia, R.A.M.: Detection and Tracking of Multiple. Partially Occluded Humans by Bayesian Combination of Edgelet based Part Detectors. Int. J. Comput. Vis. **75**, 247–266 (2007)
41. Tsukada, A., Background, A.: Road structure based scene understanding for Intelligent Vehicle Systems. 2010 IEEE/RSJ Int. Conf. Intell. Robot. Syst. 5557–5562 (2010)
42. Hu, Q., Wang, P., Shen, C., Porikli, F.: Pushing the Limits of Deep CNNs for Pedestrian Detection. Comput. Vis. Pattern Recognit. (2016)
43. Navarro, P.J., Fernández, C., Borraz, R., Alonso, D.: A Machine Learning Approach to Pedestrian Detection for Autonomous Vehicles Using High-Definition 3D Range Data. Sensors. (2017)
44. Harris, M.: New Pedestrian Detector from Google Could Make Self-Driving Cars Cheaper, http://spectrum.ieee.org/cars-that-think/transportation/self-driving/new-pedestrian-detector-from-google-could-make-selfdriving-cars-cheaper
45. Hsu, J.: Deep learning makes driverless cars better at spotting pedestrians. IEEE Spectr. (2016)
46. Xu, Y., Xu, D., Lin, S., Han, T.X.: Detection of Sudden Pedestrian Crossings for Driving Assistance Systems. IEEE Trans. Syst. Man, Cybern. Syst. 42, 729–739 (2012)
47. Peterson, K., Ziglar, J., Rybski, P.E.: Fast feature detection and stochastic parameter estimation of road shape using multiple LIDAR. IEEE/RSJ Int. Conf. Intell. Robot. Syst. 22–26 (2008)
48. Beyeler, M., Mirus, F., Verl, A.: Vision-based robust road lane detection in urban environments. 2014 IEEE Int. Conf. Robot. Autom. 4920–4925 (2014)
49. Felisa, M., Zani, P., Dipartimento, V.: Robust monocular lane detection in urban environments. 2010 IEEE Intell. Veh. Symp. 591–596 (2010)
50. Zhou, S., Gong, J., Xiong, G., Chen, H., Iagnemma, K.: Road Detection using support vector machine based on online learning and evaluation. 2010 IEEE Intell. Veh. Symp. 256–261 (2010)
51. Nair, V., Parthasarathy, N.: Supervised Learning Methods for Vision Based Road Detection. Stanford Univ. (2012)
52. Alam, F., Mehmood, R., Member, S., Katib, I., Nasser, N.: Data Fusion and IoT for Smart Ubiquitous Environments: A Survey. IEEE Access. **3536**, 1–24 (2017)
53. Xu, P., Davoine, F., Zhao, H., Denœux, T.: Multimodal information fusion for urban scene understanding. Mach. Vis. Appl. (2014)
54. Nuss, D., Thom, M., Danzer, A., Dietmayer, K.: Fusion of Laser and Monocular Camera Data in Object Grid Maps for Vehicle Environment Perception. 2014 17th Int. Conf. Inf. Fusion. (2014)
55. Premebida, C., Batista, J., Nunes, U.: Pedestrian Detection Combining RGB and Dense LIDAR Data. 2014 IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS 2014). (2014)
56. Cho, H., Seo, Y., Kumar, B.V.K.V., Rajkumar, R.R.: A multi-sensor fusion system for moving object detection and tracking in urban driving environments. 2014 IEEE Int. Conf. Robot. Autom. 1836–1843 (2014)
57. Chumerin, N., Hulle, M.M. Van: Cue and Sensor Fusion for Independent Moving Objects Detection and Description in Driving Scenes. Signal Process. Tech. Knowl. Extr. Inf. Fusion. 161–180 (2008)
58. Häne, C., Sattler, T., Pollefeys, M.: Obstacle detection for self-driving cars using only monocular cameras and wheel odometry. 2015 IEEE/RSJ Int. Conf. on Intelligent Robot. Syst. (2015)
59. Zhao, Y., Li, J., Li, L., Zhang, M., Guo, L.: Environmental Perception and Sensor Data Fusion for Unmanned Ground Vehicle. Math. Probl. Eng. **2013**, (2013)
60. Goebel, K., Yan, W.: Choosing classifiers for decision fusion. GE Glob. Res.
61. Fauvel, M., Member, S., Chanussot, J., Member, S.: Decision fusion for the classification of urban remote sensing images. 44, 2828–2838 (2006)

62. Yager, R.R.: A general approach to the fusion of imprecise information. Wiley (1997)
63. Ye, Z., Bai, L., Tan, L.: Hyperspectral image classifcation based on gabor features and decision fusion. 2017 2nd Interational Conf. Image, Vis. Comput. 478–482 (2017)
64. Bar Hillel, A., Lerner, R., Levi, D., Raz, G.: Recent progress in road and lane detection: A survey. Mach. Vis. Appl. **25**, 727–745 (2014)
65. Tsai, L.-W., Hsieh, J.-W., Chuang, C.-H., Fan, K.-C.: Lane detection using directional random walks. Intell. Veh. Symp. 2008 IEEE. (2008)
66. Li, Q., Zheng, N., Cheng, H.: Springrobot: a prototype autonomous vehicle and its algorithms for lane detection. IEEE Trans. Intell. Transp. Syst. **5**, (2004)
67. Shu, Y., Tan, Z.: Vision based lane detection in autonomous vehicle. Fifth World Congr. Intell. Control Autom. (2004)
68. Southall, B., Taylor, C.J.: Stochastic road shape estimation. Proc. Eighth IEEE Int. Conf. Comput. Vision. ICCV 2001. 1, 205–212 (2001)
69. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for Autonomous Driving ? The KITTI Vision Benchmark Suite. Conf. Comput. Vis. Pattern Recognit. 3354–3361 (2012)
70. Lieb, D., Lookingbill, A., Thrun, S.: Adaptive road following using self-supervised learning and reverse optical flow. Proc. Robot. Sci. Syst. (2005)
71. Zhou, S., Iagnemma, K.: Self-supervised learning method for unstructured road detection using fuzzy support vector machines. 2010 IEEE/RSJ Int. Conf. Intell. Robot. Syst. 1183–1189 (2010)
72. Wang, J., Ji, Z., Su, Y.: Unstructure road detection using hybrid features. Proc. 8th Int. Conf. Mach. Learn. Cybern. 12–15 (2009)
73. Huval, B., Wang, T., Tandon, S., Kiske, J., Song, W., Pazhayampallil, J., Andriluka, M., Rajpurkar, P., Migimatsu, T., Cheng-yue, R., Mujica, F., Coates, A., Ng, A.Y.: An empirical evaluation of deep learning on highway driving. Comput. Vis. Pattern Recognit. 1–7 (2015)
74. Assidiq, A.A., Khalifa, O.O., Islam, M.R., Khan, S.: Real time lane detection for autonomous vehicles. 2008 Int. Conf. Comput. Commun. Eng. 82–88 (2008)
75. Cheng, J., Mattiuzzi, M., Sumner, M., Greenberg, J.A., Bevan, A., Shortridge, A., Hijmans, M.R.J.: Raster: Geographic Data Analysis and Modeling. CRAN. (2016)
76. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
77. Wu, X., Kumar, V., Quinlan, J.R., Ghosh, J., Yang, Q., Motoda, H., Mclachlan, G.J., Ng, A., Liu, B., Yu, P.S., Michael, Z.Z., David, S., Dan, J.H.: Top 10 algorithms in data mining. Knowl. Inf. Syst. 1–37 (2008)
78. R.SDeFries, Cheung-WaiChan, J.: Multiple criteria for evaluating machine learning algorithms for land cover classification from satellite data. Remote Sens. Environ. 74, (2000)
79. Hao, P., Wang, L., Niu, Z.: Comparison of Hybrid Classifiers for Crop Classification Using Normalized Difference Vegetation Index Time Series: A Case Study for Major Crops in North Xinjiang, China. PLoS One. **10**, (2015)
80. Yang, C., Wu, G., Ding, K., Shi, T., Li, Q., Wang, J.: Improving Land Use/Land Cover Classification by Integrating Pixel Unmixing and Decision Tree Methods. Remote Sens. **9**, (2017)
81. Mehta, S., Shukla, D.: Optimization of C5.0 classifier using Bayesian theory. 2015 Int. Conf. Comput. Commun. Control. (2015)
82. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (1999)
83. Barui, S., Latha, S., Samiappan, D., Muthu, P.: SVM Pixel Classification on Colour Image Segmentation. J. Phys. Conf. Ser. **1000**, (2018)
84. Wang, X.-Y., Wang, T., Bu, J.: Color image segmentation using pixel wise support vector machine classification. Pattern Recogn. **44**, 777–787 (2011)
85. Varma, M.K.S., Rao, N.K.K., Raju, K.K.: Pixel-Based Classification Using Support Vector Machine Classifier. 2016 IEEE 6th Int. Conf. Adv. Comput. (2016)
86. Liu, D., Chen, J., Wu, G., Duan, H.: SVM-based remote sensing image classification and monitoring of Lijiang Chenghai. 2012 2nd Int. Conf. Remote Sensing, Environ. Transp. Eng. (2012)

87. Li, J., Zhao, B., Zhang, H., Jiao, J.: Face recognition system using SVM classifier and feature extraction by PCA and LDA combination. 2009 Int. Conf. Comput. Intell. Softw. Eng. (2009)
88. Heisele, B., Ho, P., Poggio, T.: Face recognition with support vector machines: global versus component-based approach. Proc. Eighth IEEE Int. Conf. Comput. Vision. ICCV 2001. (2001)
89. Litjens, G., Kooi, T., Bejnordi, B.E., Setio, A.A.A., Ciompi, F., Laak, M.G.J.A.W.M. va. der, Ginneken, B., I.Sánchez, C.: A survey on deep learning in medical image analysis. Med. Image Anal. 42, (2017)
90. Chen, C., Li, O., Barnett, A., Su, J., Rudin, C.: This Looks Like that: Deep Learning for Interpretable Image Recognition. Cornell Univ. Libr. (2018)
91. Young, T., Hazarika, D., Poria, S., Cambria, E.: Recent Trends in Deep Learning Based Natural Language Processing. Cornell Univ. Libr. (2017)
92. Wang, L., Sng, D.: Deep Learning Algorithms with Applications to Video Analytics for A Smart City: A Survey. arXiv1512.03131 [cs]. 1–8 (2015)
93. Xie, Y., Le, L., Zhou, Y., V.Raghavan, V.: Deep Learning for Natural Language Processing. Handb. Stat. (2018)
94. Deng, L., Hinton, G., Kingsbury, B.: New types of deep neural network learning for speech recognition and related applications: an overview. 2013 IEEE Int. Conf. Acoust. Speech Signal Process. (2013)
95. Graves, A., Hinton, A.M.G.: Speech recognition with deep recurrent neural networks. 2013 IEEE Int. Conf. Acoust. Speech Signal Process. (2013)
96. Salman, A.G., Kanigoro, B., Heryadi, Y.: Weather forecasting using deep learning techniques. 2015 Int. Conf. Adv. Comput. Sci. Inf. Syst. (2015)
97. Jones, N.: How machine learning could help to improve climate forecasts. Nature. **548**, 379–380 (2017)
98. Cao, C., Liu, F., Tan, H., Song, D., Shu, W., Li, W., Zhou, Y., Bo, X., Xie, Z.: Deep Learning and Its Applications in Biomedicine. Genomics Proteomics Bioinformatics. **16**, 17–32 (2018)
99. S, M., B, L., S., Y.: Deep learning in bioinformatics. Br. Bioinform. 18, 851–869 (2017)
100. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature. **521**, 436–444 (2015)
101. Deng, L.: A tutorial survey of architectures, algorithms, and applications for deep learning. APSIPA Trans. Signal Inf. Process. **3**, e2 (2014)
102. Bengio, Y.: Learning Deep Architectures for AI. Found. Trends® Mach. Learn. 2, 1–127 (2009)
103. Wiseman, E.: Strategic Technical Insights: Deep learning for human decision support. (2017)
104. Candel, A., Lanford, J., LeDell, E., Parmar, V., Arora, A.: Deep learning with H2O deep learning with H2O. Presented at the (2015)
105. Kuhn, M., Weston, S., Coulter, N., Culp, M.: C5.0 Decision trees and rule-based models. CRAN. (2015)
106. Kuhn, M., Wing, J., Weston, S., Williams, A., Keefer, C., Engelhardt, A., Cooper, T., Mayer, Z., Brenton Kenkel, the R Core Team, Michael Benesty, R.L., Andrew Ziem, Luca Scrucca, Yuan Tang, Can Candan, and T.H.: Classification and Regression Training. CRAN. (2017)
107. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic Minority Over-sampling Technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
108. Top 500 Supercomputers
109. Smeeton, N.C.: Early History of the Kappa Statistic. Biometrics. **41**, (1985)
110. Landis, J.R., Koch, G.G.: The measurement of observer agreement for categorical data. Biometrics. **33**, 159–174 (1977)

Chapter 7
A Smart Disaster Management System for Future Cities Using Deep Learning, GPUs, and In-Memory Computing

Muhammad Aqib, Rashid Mehmood, Ahmed Alzahrani, and Iyad Katib

7.1 Introduction

Smart cities appear as "the next stage of urbanization, subsequent to the knowledge-based economy, digital economy, and intelligent economy" [1, 2]. Smart cities aim to "not only exploit physical and digital infrastructure for urban development but also the intellectual and social capital as its core ingredient for urbanization"[1, 2]. Smart cities are driven by, or involve, integration of multiple city systems, such as transport, healthcare, and operations, and hence are considered a major driver for the transformation of many industries [2, 3]. Smart society is an extension of the smart cities concept, "a digitally-enabled, knowledge-based society, aware of and working towards social, environmental and economic sustainability" [2]. A recent book has covered a number of topics related to smart cities and societies [4].

Smart cities rely on dynamic monitoring and management of city assets and systems and this generates data [5, 6] of diverse characteristics, known as big data. Formally, big data refers to the "emerging technologies that are designed to extract value from data having four Vs characteristics; volume, variety, velocity and veracity" [7]. Big data leverages distributed and high performance computing (HPC) technologies to manage and analyze data. These two technologies (big data and HPC) are converging to address their individual limitations and exploit their synergies [8].

M. Aqib · A. Alzahrani · I. Katib
Department of Computer Science, FCIT, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: mpervez@stu.kau.edu.sa; asalzahrani@kau.edu.sa; iakatib@kau.edu.sa

R. Mehmood (✉)
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: RMehmood@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_7

Smart cities must be equipped with disaster and emergency management systems to manage manmade and natural calamities such as floods, hurricanes, earthquakes, fires, and terrorist attacks. Disasters not only result in loss of human lives but could also damage the economy. For example, the June/July 2018 floods in Japan left around 200 people dead and many injured. Millions of people were ordered to evacuate the affected areas, and thousands were transferred to temporary shelters [9, 10]. Rescue teams comprised of workers from civil defense and many other organizations. They worked round-the-clock to overcome the disastrous situation in the affected areas. The cost of flood rebuilding was estimated to be $2bn [11]. The Barcelona terrorist attack of August 2017 resulted in 24 deaths and 152 injured people. Earlier, the cost of 2011 Japan earthquake and tsunami disaster alone was in excess of 200 billion USD in addition to the irrecoverable loss of over 18 thousand lives [12].

The advent of many new technologies has improved our ability to manage disaster situations. Many governments around the world are applying these new techniques and technologies to minimize the effects of these disasters. Plans are made to respond not only during the disaster situation, but also after the disaster, and more importantly, how to prevent or minimize the effects of disasters before its occurrence.

Mobility plays a key role in effectively managing disaster situations [13]. Smart mobility requires smart transportation infrastructures [14, 15]. Many approaches have been developed to improve transportation. These include, for example, social media based approaches [16–18], big data based techniques [15, 19, 20], HPC based techniques [15, 19, 21, 22], vehicular networks (VANETs) and systems [23–26], modeling and simulations [27, 28], methods to improve urban logistics [15, 19, 21, 29, 30], and solutions based on autonomous vehicles and autonomic mobility systems [31–34].

Smart mobility allows smooth evacuation of people from the affected areas by dynamically monitoring the disaster-affected areas as well as the other adjacent areas to avoid road congestion, blockages, and chaos. Traffic data is collected from various static and mobile sensors including those deployed under and over the road networks. These could include inductive sensors, motorway incident detection and automatic signaling (MIDAS) loops, GPS sensors, VANETs, cameras, image processing systems, and many more. The collected data is analyzed to monitor traffic flow and other metrics, and is used to devise navigation strategies to provide emergency services and smooth evacuation from the affected areas, avoiding congestion, minimizing risks to public safety, and economic losses.

Our research focuses on using emerging technologies to develop cutting edge solutions for disaster management. We have proposed a cloud computing based disaster management system along with its implementation in [13]. The work was extended in [35] leveraging VANETs to sense traffic related information and propagate navigation instructions. These works were based on macroscopic modeling. Further improvements to the disaster management system were proposed in [12] where microscopic modeling was used to improve and validate the earlier results. Moreover, different evacuation strategies were used to investigate the

performance of evacuation operations on the proposed disaster management system. Further extensions of the proposed system were reported in [36, 37] using various evacuation strategies.

The availability of various data related to smart environments, generated, for instance, by the internet of things (IoT), and the advancements in artificial intelligence (AI) has provided new opportunities for data-driven studies (see, e.g., [2, 3, 38]). Deep learning has emerged as a promising AI technology with reportedly higher prediction accuracy, albeit higher computational costs [39]. In [22], we extended our work by using deep learning to predict traffic plans for evacuation in disaster situations. We had used in-memory computations and graphics processing units (GPUs) to address intensive and timely computational demands in disaster situations.

This paper extends our earlier work and provides extended analysis and results of the proposed system. A system architecture based on the in-memory big data management and GPU-based deep learning computations is proposed. The background technologies have been elaborated. An extended literature review is provided. We have used road traffic data made publicly available by the UK Department for Transport (DfT). The results show the effectiveness of the deep learning approach in predicting traffic behavior in disaster and city evacuation situations. To the best of our knowledge, this is the first proposal where deep learning, in-memory data-driven computations, and GPUs are brought together for timely, compute intensive, predictions of road traffic in disaster situations.

The rest of the paper is organized as follows: Sect. 7.2 provides background material introducing the tools and technologies used in our work. The related work is discussed in Sect. 7.3. Our proposed framework is introduced in Sect. 7.4. In order to find suitable city data, we have examined a number of datasets and their details are given in Sect. 7.5. These could be useful for the researchers working in related areas. Performance evaluation and analysis of the proposed system is given in Sect. 7.6. Finally, in Sect. 7.7, we conclude the paper with directions for the future work.

7.2 Background Material

In this section, we will give a brief introduction to the tools and technologies used in our model in specific and some tools and simulators that are used for traffic modeling in general.

7.2.1 Graphical Processing Units

In this section, we will give an overview of the GPU architecture. A GPU chip contains multiple multi-processors (MPs) and each MP contains many stream-processors (SPs). Instructions are executed in SP like ALU in CPU. Different tasks

are performed on MPs and they are mutually independent to each other, whereas the SPs in an MP execute the same operations on different data items. To store data, each SP has its own register to store variables and temporal data. An SP cannot access the registers of other SPs in an MP. For this purpose, there is a shared on-chip memory that is accessible to each SP in that MP. In addition to this, an off-chip shared memory, called global memory is also available and it can be accessed by all the SPs in all the MPs. This global memory is connected externally to the GPU chip and it is much larger in size but the access to this memory is much more expensive than that of the on-chip shared memory inside the MPs.

Programs in GPU are executed with the help of compute unified device architecture (CUDA) toolkit offered by Nvidia and detailed execution flow of a CUDA, the logical structure of kernel threads, and logical to physical mapping in GPU are also part of the discussion.

7.2.2 In-Memory Computing

For computation purposes, data is normally stored on disks that provide the facility to store a large amount of data. In addition to disks, other memory storage components are also used for this purpose that include registers, cache, and main memory. These storage components differ in size and also in performance. Registers are the smallest ones in terms of capacity to store data but these are the most efficient in terms of speed. Then there are caches and main memory in this hierarchy, which are much smaller than disks but provide higher speed to access data. In recent years, the main memory size has also been increased and its cost is also decreasing that make it possible to use large amount of main memory to perform compute intensive tasks. Due to increased size, it is capable to hold a large amount of data as well, thus making it easy for the programs to access that data on low I/O costs. In-memory computing also supports the technique to store the data required for processing on the main memory instead of storing it on the disks. This idea was introduced a long ago but high price and availability of low storage capacities were the constraints in using in-memory technique to deal with large amount of data. Now using in-memory, a large amount of data could be stored into the main memory for processing. In case of big data, where data size is large enough so that it could not be stored in main memory of a single computer, it is normally distributed among multiple nodes in a cluster of compute nodes and each node is assigned a block of data according to its capacity. Many frameworks exist that distribute the data to all the nodes in the cluster to be stored on the main memory and then processed. The results generated by the individual nodes are then combined to generate unique output.

Due to increase in cost of energy and increase in its demand as compared to the production rate, researchers are now finding the ways to optimize the existing systems or methods to develop the new energy efficient systems. The authors in [40] have studied the role of database software in order to improve the efficiency

of a server. According to them, among the nodes in a scale-out architecture, the highest performing one is considered as the most energy efficient configuration. Also the power consumed by different operators like joins, sorts, etc., varies and also the CPU power consumption and its utilization are not linearly related to each other. In a survey of the energy efficiency techniques [41] have focused on the characteristics of the two main power management technologies: static power management (SPM) systems and dynamic power management systems (DPM). The article presents a brief discussion on the techniques proposed by researchers to reduce the power consumption in cluster computing systems. The pros and cons of both the methodologies have been discussed in detail. Non-volatile memory has great importance in main memory data management systems. But it has many issues as well and energy consumption is one of them. A technique has been proposed in [42] that deals with the high energy consumption rate during write operations in phase change memory (PCM). A solution based on out of position PCM write operations has been proposed that reduces power consumption, however, degrades the system performance. PDRAM [43] is another approach for in-memory data management systems based on the phase change random access memory (PRAM) and DRAM. The authors have proposed an approach that deals with the low read and standby power and DRAM has low write power by providing a hybrid hardware software solution. Some other techniques that do not suggest the storage of whole data in main memory also propose a mechanism to store the data needed for computation in main memory. Such a technique [44] proposes the bulk copy and initialization completely in the DRAM, which in return reduces the data transfer over the memory channels and thus saves energy. The proposed technique is named as RowClone and it copies the complete row of data from source to a row buffer and then from the buffer to the destination. As part of semi-structured data processing, SAP HANA provides the facility to process graphs data.

7.2.3 Deep Learning

A branch of computer science that gives the computers the ability to learn themselves like human beings is known as machine learning. Machine learning does not require programmers to program something explicitly to tell computers to perform a specific task. Instead, machine learning algorithms train computers using different algorithms to predict the output when a specific input is given. Techniques that enable computers to learn something without explicit programming are divided into two main categories in machine learning. These are known as supervised learning and unsupervised learning techniques. Artificial neural network, clustering, genetic algorithms, and deep learning are some examples of machine learning techniques. In this section, we will focus on the deep learning techniques and work done in this domain.

Deep learning approaches have been classified into different categories based upon the nature and training and testing strategies. These include convolutional

neural networks (CNNs), restricted Boltzmann machines (RBMs), autoencoders, and sparse coding techniques [45]. In this work, we are using CNNs for training and testing purposes. So, we will discuss them in detail in the following paragraph.

Convolutional Neural Networks (CNNs) In the CNNs, multiple layers including convolutional, pooling, and connected layers are used for training purpose in a robust manner. The authors in [45] have defined a general architecture of CNN for image classifications. The whole process is divided into two main phases: forward phase that includes convolutional and pooling layers and backward phase where fully connected layers are used to produce the output.

Convolutional neural networks are the hierarchical neural networks and their convolutional layers alternate with subsampling layers like simple and complex cells in the primary visual cortex. CNNs vary in how convolutional and subsampling layers are realized and how the nets are trained [46].

7.2.4 Microscopic Models and Tools

In this section, we will discuss the microscopic model that is used in traffic management works. Although, instead of using these models or any other related simulation tools, we are using deep learning to forecast traffic plans in disaster situations but here we are giving a brief introduction about other techniques to give an overview of these models to the readers.

Lighthill–Whitham–Richards (LWR) model [47, 48] is a macroscopic model that could be used to analyze the traffic behavior in roads. It uses some traffic data characteristics like speed, flow, and density. This model could be derived from the following equation:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = 0 \tag{7.1}$$

Here ρ is the traffic density, x is the distance, t is the time, and u is the speed to travel x distance in t time. Now using the Greenshields' model [49], the relation between the density (ρ) and speed (u) could be defined as follows:

$$u(\rho) = u_{max} = \left(1 - \frac{\rho}{\rho_{max}}\right) \tag{7.2}$$

Here u_{max} is the maximum speed, and ρ_{max} is the maximum density. So, by using this model, the relationship between flow, density, and speed could be given as

$$\text{flow} = \text{density} \times \text{speed} \tag{7.3}$$

To model these microscopic models, a number of simulators are available for this purpose. In the following paragraphs, we will discuss about two of them which are used by researchers to carry out their research work. MITSIM [50] is a microscopic traffic simulator that uses the information related to road network, surveillance system, traffic signs and signals, etc. It classifies the lanes according to its speed limit, regulations, and also simulates loop detectors, lane use signals, etc. As an input, an origin–destination table, traffic control, and route guidance logic are used. Here vehicles are considered to move between their origin and destination and it collects the sensor readings that include traffic count, occupancy, and speed of vehicles at given intervals of time. To simulate the vehicle movement in a network, two models are used that are: acceleration model and lane changing model [51].

Simulation of urban mobility (SUMO) [52] is another microscopic and continuous road traffic simulation package that deals with the large road networks. It provides the users the facility to define their own network through the use of data configuration files. Normally input data is given in the form of XML files where different nodes having different parameter values define different configuration values. It also provides the facility to generate real world scenario by selecting the area on a map in a browser by running a program included in package.

7.3 Related Work

In this section, we are presenting the work that deals with the traffic management plans during emergency conditions in smart cities. Some people focus mainly on traffic management in smart cities using any approach and some have focused on the approach, i.e., deep learning with smart city scenario on low priority. As we are combining traffic management in smart cities with the deep learning approach, both are useful for us and therefore we are presenting some approaches for better understanding of the work done in this area.

A deep learning approach to predict traffic flow for short intervals on road networks is proposed in [53]. A traffic prediction method based on long short-term memory (LSTM) has been used by the authors for prediction purpose. An origin–destination correlation (ODC) matrix has been used as input to the training algorithm. Dataset used for this process is collected from the Beijing Traffic Management Bureau and it is collected from more than 500 observation stations or sensors containing around 26 million records. A 5-min interval data from Jan 1, 2015, to June 30, 2015, has been collected where the data for the first 5 months has been used for training and the rest of the data is used for testing purposes. For evaluation of proposed model, mean absolute error (MAE), mean square error (MSE), and mean relative error (MRE) have been calculated. Input data has been used to predict the flow in 15, 30, 45, and 60 min time intervals. The authors in this work have selected three observation points with high, medium, and low flow rates to compare the actual flow and predicted flow values on those observation points. MRE values for a 15-min interval flow prediction reported in this work are

6.41, 6.05, and 6.21%. They have compared the result with the other approaches including RNN, ARIMA, SVM, RBF, etc., and concluded that for time interval less than 15 min, RNN is relatively accurate, but with big time intervals, error increases, but overall it performs better than other old machine learning models. Therefore, it is concluded that LSTM is an appropriate choice for long time intervals.

Yu et al. in [54] also have proposed an approach that uses deep learning for vehicles' speed prediction on highways in peak hours and post-accident conditions. In this work, the authors have used the long short-term memory (LSTM) recurrent neural networks for prediction purposes. In addition to LSTM, they also have used autoencoders whose output is also used in their deep LSTM model. This model is named as "mixture deep LSTM". For this purpose, they have used the data from the 2018 loop detectors (sensors) in Los Angeles County during the period starting from May 19, 2012, to June 30, 2012. This provides data collected from around 5400 miles long highways cumulatively. Also, as they are predicting the speed after accidents as well, so accidents data, for this purpose, has been collected from various sources including California Highway Patrol, California Transportation Agencies, etc. Normalized data including 5 min aggregated speed values, day time, and weekdays has been used in this work. To deal with the missing values, data collected from the sensors with more than 20% missing values is excluded from the datasets. Also, no criteria is defined to deal with the missing values and the estimated speed values for missing input values have been excluded from the predicted output datasets and have not been considered for evaluation. For analysis purposes, mean absolute percentage error (MAPE) has been used. Performance of the proposed is compared with other methods like ARIMA, random walk, and historical average, etc. Speed in peak hours has been predicted using four different time intervals of 5, 15, 30, and 60 min. The results show that the highest accuracy is achieved for small time interval, i.e., 5 min. It is around 6 in peak hours and around 5.5 in off-peak hours. Error rate increases with the increase in time interval but in all the four intervals, deep LSTM performs much better than other techniques.

In another work, Jia et al. have used a deep learning approach called deep belief networks (DBN) to predict the vehicles' speed on a road network in [55]. In this work, they have used restricted Boltzmann machines (RBMs) for unsupervised learning and then have used the labeled data for fine tuning. Dataset used in this purpose is obtained from Beijing Traffic Management Bureau (BTMB). Three months data (June–August 2013) has been used that provided 2-min interval data collected from the detectors installed on a specified segment of road in Beijing, China. Around 11-week data is used for training purpose, whereas the remaining last week's data is used for testing purpose. This provides 2-min interval speed, flow, and occupancy values and by using this 2-min interval data, the authors have predicted speed for intervals of 2, 10, and 30 min. Furthermore, for performance analysis, three performance metrics have been used: mean absolute percentage error (MAPE), root mean squared error (RMSE), and normalized root mean squared error (RMSN). No mechanism is mentioned by authors to deal with the erroneous or missing data values and also no big data technology is used for data management. Also, no specific information about data, e.g., number of detectors, etc., is given to know

about the size of data. For deep model configurations, they have executed the model with different configuration and based on the MAPE values, best configurations have been selected. With best selected configurations, MAPE value for 2-min interval is 5.81, 7.33 for 10 min, and its value is 8.48 for 30-min interval. This shows that it performs better for short time intervals and cannot cope with the stochastic fluctuations in long time intervals. Although results are quite good for speed prediction, but it still need to investigate how it behaves when some other information are included, e.g., we do not know whether data from multiple detectors has been used or separate data for each detector is used because in the former case we get more fluctuations in data as compared to the latter case. Also, the size of data could also change the results.

The authors in [56] have proposed an adaptive traffic management plan to ensure the provision of secure and efficient emergency services in case of disaster in a smart cities. In this work, a framework has been proposed, which introduces some components of traffic management system like traffic management controllers (TMC), local traffic controllers (LTC), adaptive traffic light controllers, environmental sensor controllers, etc. The goal of this framework is to collect information from communication and other devices about the severity of the disaster that has been divided into three categories in this work: low, medium, and high, and then act accordingly by using these controllers. For example, in case of high emergency condition, traffic signals could be controlled to ensure the timely arrival of emergency vehicles, e.g., ambulance and fire brigade and to reroute the non-emergency traffic. SUMO [52] has been used to simulate this process. In this work, focus is mainly on the provision of emergency services and their security and the plan has been simulated but no practical scenario or data has been used to handle the traffic and it also lacks the plan to manage the general traffic in case of disaster.

Smart cities are characterized by advanced and integrated ICT systems, such as smart logistics solutions [16] and autonomic transportation [31]. Internet of things (IoT) could be considered as the back bone of future smart cities [38]. Mehmood et al. [2] propose a ubiquitous learning system for smart societies. This approach can be used to educate and prepare citizens for disasters. In particular to vehicles, internet of vehicles (IoV) includes all the devices that could be used to monitor the vehicles and for inter-vehicle communication as well. Data from different types of sensors placed on road networks, vehicles, and other smart devices [1] is collected for traffic management. There are many studies that use IoT and IoV to propose a traffic management plan as in [57, 58]. In addition to this, a lot of work has been done in the area of autonomic transport management in smart cities [33]. The work in [30] also shows the importance of fog and other cloud technologies in dealing with emergency situations in smart cities. In [59] a parallel transportation management and control system for smart cities has been presented that not only uses the artificial intelligence technologies but also uses massive traffic data and big data technologies or frameworks like MapReduce. This shows the importance of these technologies in traffic management in smart cities.

A traffic flow prediction approach has been proposed in [60]. The authors have used the deep learning approaches for prediction purpose using a large amount of

data. They have proposed a model that uses autoencoders for training and testing purpose to make predictions. The model is named as stacked autoencoder (SAE) model. To predict traffic flow at time t, traffic flow data at previous time intervals has been used. The proposed model has been used to predict 15, 30, 45, and 60 min traffic flow. Data for this purpose was collected from Caltrans Performance Measurement System (PeMS) [61]. Three months data collected every 30 s was used for training and testing purposes. In this data, vehicle flow was collected where two directions of the same freeway were treated as different freeway. Support vector machines (SVM) have been used for comparison purpose.

The authors in [62] have proposed a deep learning based approach for traffic flow prediction and they have used unsupervised learning approach using deep belief networks. They have categorized the traffic prediction approaches into three main categories that include time-series approaches, probabilistic approaches, and non-parametric approaches such as neural network based approaches, etc. The authors in this work have used restricted Boltzmann machines (RBMs) for training purpose which are stacked one on other. For training and testing purposes, inductive loop dataset is obtained from the PeMS [61]. In addition to this, the authors have used data from highway system of China (EESH) as well. A data of 12 months has been collected and the first 10 months data is used for training, whereas the data of remaining 2 months has been used for validation purpose. Prediction results have been compared with other four methods for top 50 roads having high flow rates. The results show that deep learning based architecture is more appropriate and robust in prediction and could be used for practical prediction system.

A deep learning based approach has been used in [63] to model the traffic flow. In this work, the authors have developed deep learning predictors to predict the traffic flow data from the road sensors. Real-time traffic data has been used and by using the proposed model, they have predicted the traffic flow during a Chicago Bears football game and a snowstorm. They have used the number of locations on the loop detectors and traffic flow at a time (say t). They first have developed a linear vector autoregressive model for predictors selection. These predictors are later used to build a deep learning model. Stochastic gradient descent (SGC) method is used to know the structure and weights of parameters. They also have applied three filtering techniques (exponential smoothing, median filter, and loess filter) on traffic data to filter noisy data from the sensors. Data for this purpose is collected from 21 loop detectors on 5-min interval basis. This data includes speed, flow, and occupancy. They have built a statistical model to capture the sudden changes from free flow (70 mph) to congestion (20 mph). In case of bottlenecks, they predict that how fast it will propagate on the network, i.e., loop detectors. For predictor selection, deep learning model estimates an input–output map with the assumption that they need the recent. So, they collect the last 12 readings from each sensor. The performance of DL model has been compared with sparse linear vector autoregressive (VAR). Both accurately predict morning rush hours on normal day but VAR miss-predicts congestion during evening rush hour. On the other hand, DL predicts breakdown accurately but miss-estimates the recovery time.

The authors in [64] also have used deep learning approach to predict the traffic congestion. They have used recurrent neural networks by using restricted Boltzmann machine (RNN-RBM). For comparison purposes, the authors have used support vector machines (SVMs) and found that prediction accuracy was increased by at least 17%.

7.4 Disaster Management System

In this section, we will discuss the proposed deep learning based disaster management system in detail. Figure 7.1 depicts the architecture of our proposed system. The proposed framework consists of three main layers: input layer, data processing layer, and prediction layer. A general framework was given in our previous work [22] as shown in Fig. 7.2. In this work, we have presented the complete architecture that gives details about each layer and the components in each layer. In the following sub-sections, we will discuss these layers in detail.

7.4.1 Input Layer

Input layer manages the traffic data that is used for training and testing of deep learning model in the data processing layer. The input data could be either offline, i.e., historical data, or it could be real-time or streaming data. The role of input layer is to collect data from the source and to forward it to processing layer. In case of

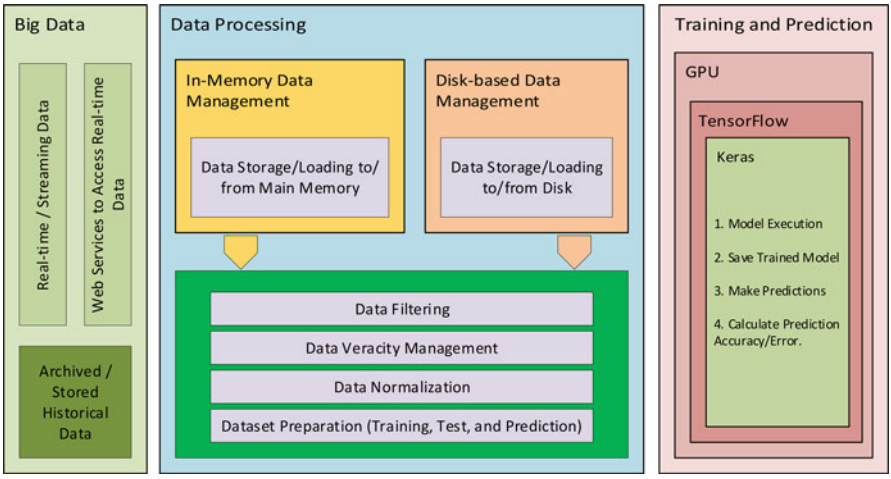


Fig. 7.1 System architecture for prediction of traffic plan using deep learning

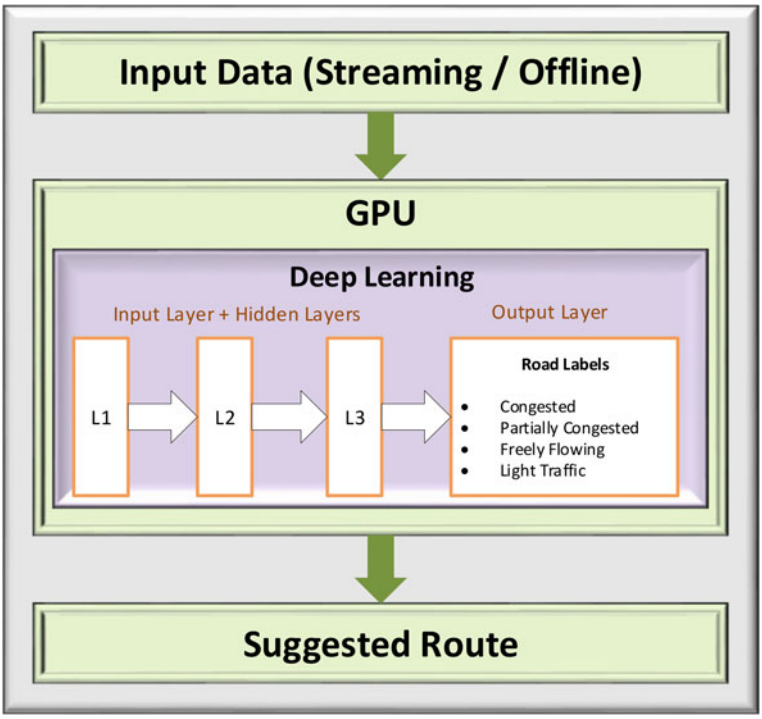


Fig. 7.2 The proposed disaster management framework

offline or historical data, the data is collected from the source and then stored on a disk drive so that it could be forwarded for processing layer. The role of input layer becomes more important especially when we are dealing with the real-time data. In this case, it takes the data from the source, by using the APIs provided by the data generating source or web services, and forwarded it to the processing layer in real-time for further data formatting.

7.4.2 Data Processing Layer

This layer is responsible to process the input data for making predictions in case of disaster. Our prediction model uses deep learning approach for this purpose. By using a deep regression model, we train a dataset which is further tested using another input dataset or a subset of the same dataset. Data processing layer takes the data from the input layer and then processes it to convert the input data into the format required by the deep learning algorithm. For example, if date attribute is included in the input dataset, it could be processed in this layer to get day, month, year, hour, etc. The division of one attribute into multiple attributes could

be useful in training process, e.g., we can get peak hours, and can separate the data based on weekends, etc. Different big data related issues like dealing with data veracity are also resolved in this layer. Because the data collected from the sensors or other devices is not guaranteed to be free of veracity issues. For example, due to malfunctioning in recording device, the recorded values may be incorrect, or missing, etc. So in this layer, we have a mechanism to deal with the erroneous data. For this purpose, well-known techniques are applied to ensure the correctness of data. Furthermore, we may need to normalize the input data for our regression model. So, data normalization is also performed in this layer.

7.4.3 Deep Learning Layer

We have used deep regression model to estimate the vehicle flow value by using multiple input features. Initially we have trained our neural network by adding two hidden layers to the network. First layer is our input layer and the final one is the output layer and the two hidden layers are in between the input and output layers. Forward propagation scheme has been used for computation of weights and finally loss is calculated on the overall output.

Figure 7.3 shows a neural network including one input, two hidden, and one output layer. In our case, we are using 9 input parameters, and output layer gives

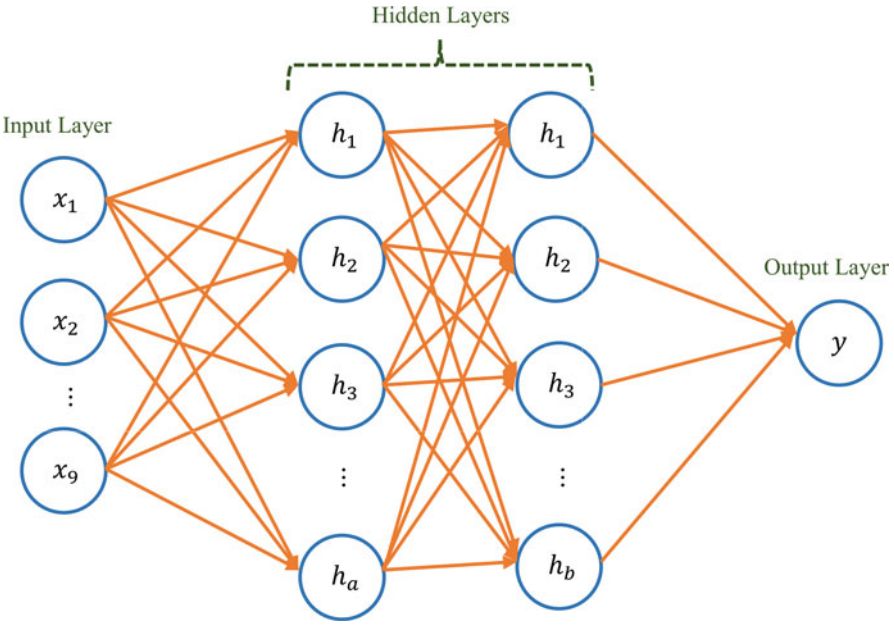


Fig. 7.3 Our deep neural network with two hidden layers

one output value because we are applying regression to get one vehicle flow value. We have used $ReLU$ activation functions and $AdamOptimizer$ has been used to optimize the generated results. We ran the training process for 1000 times by selecting a data size of 500 features at one time.

7.5 Datasets

In this work, we are mainly working on the UK traffic data. So, we have explored a variety of traffic data available through multiple sources in the UK that could be used for different purposes to work on traffic management plans. Some data sources of same kind outside the UK are also included in the list. In our deep learning model, we have used the data from data.gov.uk. that provides the vehicles flow data for minor cities. This includes the average vehicle count or roads for different vehicle types. In Table 7.1, we have given some data sources that provide traffic data. Short data description and URLs to access the data are also given.

7.6 Analysis and Comparison

This section defines our deep model configurations and the performance metrics used for analysis purpose which is used for performance analysis of our model.

7.6.1 Deep Model Setup

In this work, we have used vehicles flow data on minor roads in a city in the UK. It includes six different vehicle categories ranging from cars or small personal vehicles to big trucks used for transportation of goods. Data used as input contains 70,470 data flow values for all six vehicle categories for the years from 2000 to 2015 and the road names along with the road categories are also given.

We are using a deep regression model to predict the vehicle flow values. We have implemented this model using Keras deep learning library [65] which uses TensorFlow library [66] at the backend. Our regressing model has four layers including one input, two hidden, and one output layer. We have used the annual average flow data to predict the traffic flow in a city. Input dataset is divided in the ratio of 7, 2, and 1 for training, testing, and prediction purposes, respectively. Batch size was set to 10 and number of epoch was set to 1000.

Table 7.1 The UK traffic data sources

S.No	Data source	Description
1	Transport for London (TFL)	Data could be accessed by using the provided API. Real-time data and status information of different sources of transportation could be accessed by using API. https://tfl.gov.uk/info-for/open-data-users/
2	London Datastore	Public data sharing portal that provides data related to different department of London government. Data from 1997 to 2015 is also available that provides number of vehicles on different roads in London. https://data.london.gov.uk/
3	Data.gov.uk	Data provided by different UK government agencies could be accessed from this portal. Its transport data section provides many options to explore traffic data. https://data.gov.uk/dataset/gb-road-traffic-counts
4	Data from Local Government Association UK	This is a research project and its purpose is to make data useful for LGA. http://www.local.gov.uk/web/guest/research/-/journal_content/56/10180/7783953/ARTICLE
5	Transit Feeds	It provides web feeds for transport data and provides updated information related to transport department of a city or state, etc. http://transitfeeds.com/
6	Department for Transport UK	It provides data for all the A class roads at city level. Data collected from data collection points on roads that fall in the selected city could be accessed from this source. http://data.dft.gov.uk/
7	Transport Infrastructure Ireland (TII)	This site also provides traffic data for main roads (highways). It could be useful while dealing with the intercity traffic data. Do not provide enough data to deal with the traffic on minor roads in a city. https://www.nratrafficdata.ie
8	Tyne and Wear region data	We can access the live traffic data by using the API provided by the "Open Data Service" authority. http://www.gateshead.gov.uk/Parking-roads-and-travel/planning/TADU.aspx
9	The WisTransPortal System	Hourly traffic data index page could be accessed to get a list of counties in the Wisconsin State, USA or county could be selected from the map as well. By selecting the county, it displays all the data available for different roads in that county by their names. https://transportal.cee.wisc.edu/products/hourly-traffic-data/
10	Wisconsin Department of Transport	Provides traffic flow data on weekly and/or annual basis on selected roads (say highways). http://wisconsindot.gov/Pages/projects/data-plan/traf-counts/default.aspx
11	North East Combined Authority	Provides data for selected areas. It provides data related to special events, roadworks, incidents, journey times for key roads, car parks, and CCTV images. https://www.netraveldata.co.uk/
12	Highways England	Provides three types of data: monthly summary data, journey time data, and traffic flow data. HE also provides a conversion table that gives description of traffic data measurement sites. http://tris.highwaysengland.co.uk/
13	Website Developer.here.com	Provides API to get traffic flow and incidents data. https://developer.here.com/

Table 7.2 Schema of dataset used as input in our deep learning model

S.No	Attribute name	Description
1	Road	Gives character code names assigned to a road in the city
2	Road name	Name of the road
3	RCat	Roads have been divided into different categories. RCat gives character codes to define its category in city road network
4	iDir	Traffic direction on a road, e.g., heading east or west
5	Year	Year for which AAFD was collected
6	dCount	Day of the year when data was collected. It is in the format dd-mm-yy h:mm
7	Hour	Hour of the day
8	CAR, BUS, LGV, HGVR2, ...	A set of different types of vehicles to provide their flow values. For example, car gives the annual average flow value for cars. Similarly, bus provides the annual average flow value for buses and so on

7.6.2 Input Dataset Schema

Dataset we have used in this work contains annual average flow data for different types of vehicles. It also provides road names, road category, and other information. In Table 7.2, we have given the schema of input dataset that provides brief description of some important input attributes in this dataset.

7.6.3 Performance Metrics

For performance analysis, we have used mean absolute error (MAE) and mean absolute percentage error (MAPE). MAE is used to show the closeness between the actual and the predicted values and MAPE shows the relative difference between the actual and the predicted values. MAPE is not suitable to calculate error rate if the input data or actual values contain zeros because in this case it suffers from the division by zero error. MAE and MAPE values are calculated by using Eqs. (7.4) and (7.5), respectively.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|V_i - P_i| \tag{7.4}$$

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|V_i - P_i|}{V_i} \tag{7.5}$$

Here N is the size (number of values predicted by the model) of dataset used for prediction purpose, V is the set of actual values used as labels, and P is the set of values predicted by our deep learning model.

7.6.4 Performance Analysis

In this paper, our focus is mainly on providing details of the deep learning based traffic prediction approach. Details of the overall evacuation method can be found in our earlier work [12, 13, 22, 35]. We have executed our deep model with different configurations and with different input dataset sizes. Furthermore, we have divided the analysis process in different phases where we have used different model configurations and different dataset distribution sizes to compare the results for analysis purpose.

In the first phase of analysis process, we divided the dataset into three parts where 70% data was used for training, 20% data for testing purpose, and the rest 10% data is used for prediction purposes. We have reserved the data for prediction purpose, because, after running the model for training and purpose, we saved the model and the specified amount of data was used as input to the saved model to predict the output. This enabled us to compare the results produced by analyzing the testing dataset and the results calculated by us by analyzing the values produced by the saved model by using the prediction dataset. In addition to this, our deep learning model with one configuration setup was executed for 20 times to get results for analysis purpose. Furthermore, for all the 20 models with the same configurations, the batch size for training purpose was 10 and the training procedure was repeated for 2000 times in each execution.

We have used annual average vehicle flow data on different roads in a city to predict flow values on minor roads in a city in the UK. We have evaluated the results of all 20 executions of our model to see the variation in the accuracy and error rate. This gives a better idea about the performance of deep learning model and we calculate the average accuracy rate.

In Fig. 7.4, we have shown the results obtained by executing our deep model 20 times. In this graph, x-axis shows the number of model and it ranges from 1 to 20, and y-axis shows the MAE values calculated by using the given equation. Graph

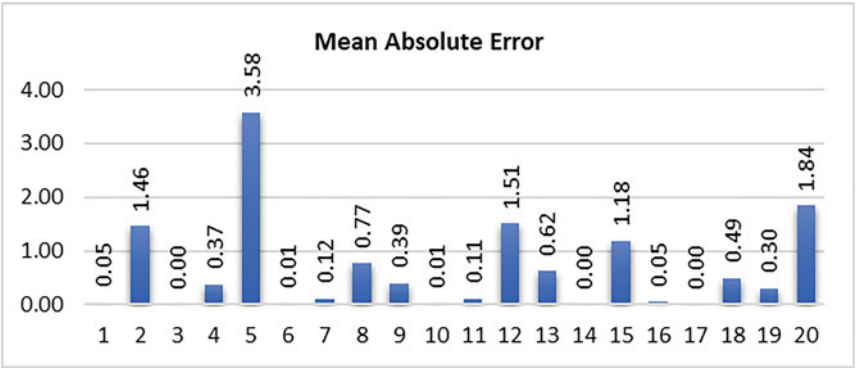


Fig. 7.4 Mean absolute error

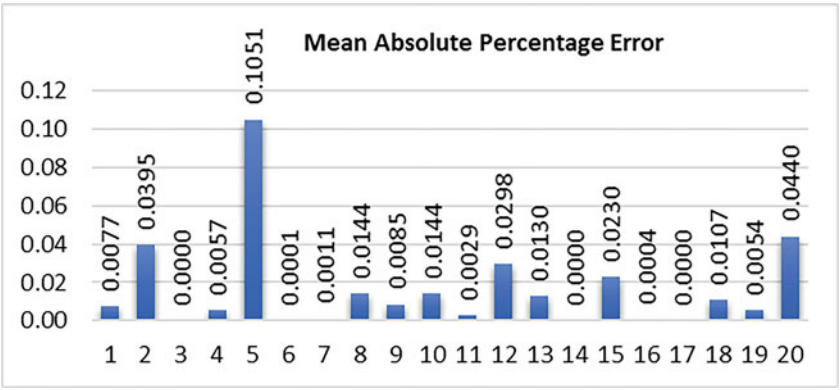


Fig. 7.5 Mean absolute percentage error

shows that error rate was very low because the maximum error value calculated was for model 5 and it was 3.58, and in some cases, it was as low as zero. Here zero does not mean that prediction was exactly the same, but it shows that the values were very close and there was not a big difference between the original and the predicted values.

In Fig. 7.5, we have shown the results calculated by using the mean absolute percentage error. Same as MAE, we have calculated MAPE for all 20 executions and prediction results of our deep learning model. Maximum MAPE value is 0.105 for 5th execution of our model with the same configurations. MAPE is considered a best measure to the data where there are no extremes and our data also contains a relatively balanced set of flow values. Therefore, our MAPE values describe that the predicted results have very low error rate and predicted values are very close to the original flow values.

In addition to the graphs showing error rates using MAE and MAPE, we have plotted the actual and predicted flow values to show the difference between patterns as well. Our MAE and MAPE values show that the actual and predicted values are very close. If this is true, then the graphs of both plotted values should show the similar trends. In Fig. 7.6, we have plotted the first 100 actual and the predicted flow values. In this graph, y-axis shows the flow values. As both, actual and predicted values are very close, graph is drawn by doubling the predicted values to avoid the overlapping of both curves. Both the curves show that these are not same but follow a similar trend. This shows that the predicted values are following the same trend that was followed by the input flow data with slight differences.

Similarly, to analyze the pattern in depth, we have selected a range of actual flow values from 1 to 500, i.e., we have selected only those results where actual flow values are in the range of 1–500. The purpose of selecting this range is to see the trends when flow values were uniform and thus input data values were very close. This is shown in Fig. 7.7. Again, the predicted values are doubled to avoid overlapping of both curves representing the flow values. This graph also shows

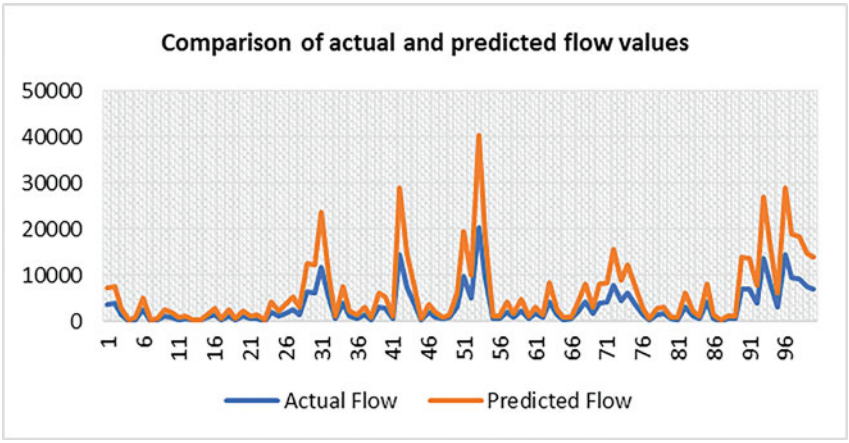


Fig. 7.6 Comparison of first 100 actual and predicted flow values

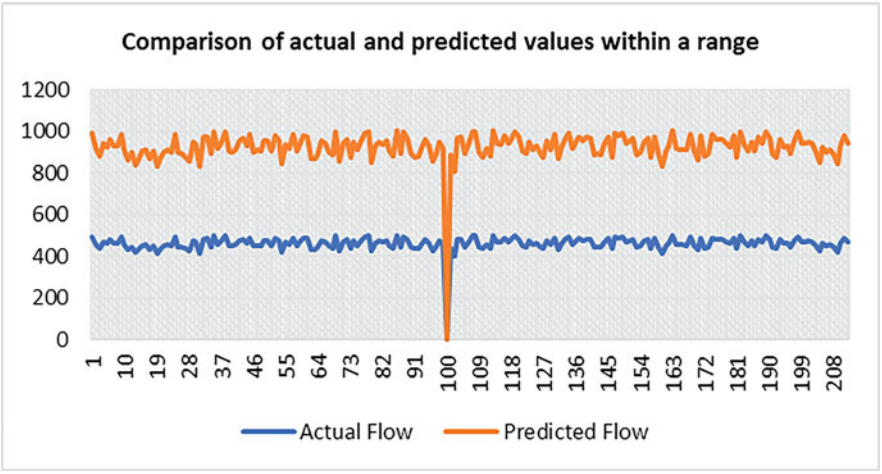


Fig. 7.7 Comparison of actual and predicted values when flow is less than 500

similar graph for both, actual and predicted flow values with not big differences. In this graph, we have selected values within a range; therefore, it is expected for good prediction results that the output values should also be in a specific range as shown in this graph. So, we can say that predicted values have followed the trend that was present in the input dataset. Therefore, the accuracy rate is high and low MSE and MAPE rates are reported.

To show the accuracy of our predict results, we also have compared the actual and predicted values. We have calculated the maximum difference between the actual and the predicted values. The main purpose to calculate the maximum difference between the actual and the predicted values in each model execution is that it clearly

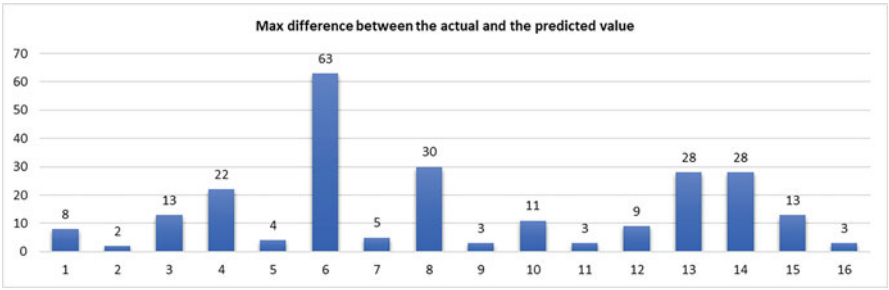


Fig. 7.8 Maximum difference between the actual and predicted vehicles flow values (phase 1)

shows whether the predicted values predict the number of vehicles that match the ground reality or it is far away from the actual values. Maximum difference between the actual and predicted vehicles flow values in each model execution is shown in Fig. 7.8.

In this figure, we have compared the available predicted values for the 16 executions of same deep model on the same input dataset. From this graph, it is clear that the minimum value for the maximum difference is 2, which shows that results obtained in this model execution were very close to the original values and we can say that it can be used to represent the actual data. On the other hand, the maximum value while calculating the maximum difference is 63, which can be used to represent the actual values if the actual vehicles flow value was very big, e.g., say 1000, but if in actual, there were only 100 vehicles on the road, then the difference of 63 between the actual and the predicted values represents the inaccuracy of predicted results that cannot be used to represent the actual values.

In this phase the distribution of the dataset for training and testing processes was changed to 60% and 30%, respectively, whereas the rest 10% was used for prediction purpose. Batch size in training process was also same, i.e., 10 but now number of iterations to repeat the training process was reduced from 2000 to 1000.

As the same procedure was repeated with different dataset sizes and iterations in training process, we have measured the same attributes for comparison purpose as we have done before. To compare the results with the previously used model configurations and the dataset distribution for training, testing, and prediction, we are again comparing the maximum difference between the actual and the predicted flow values as shown in Fig. 7.9.

From Fig. 7.9, we can see that the minimum maximum difference value is 0, and the maximum value for maximum difference between the predicted and the actual value is 38. To see whether there is overall improvement in the prediction or not, we have calculated the average maximum difference in both the cases. For first phase (Fig. 7.8), the average difference value is approximately 15, whereas it is 11.5 in phase 2 (Fig. 7.9). So, we can say that in phase 2, the accuracy as compared to the model configurations in phase 1 has improved. In addition to maximum

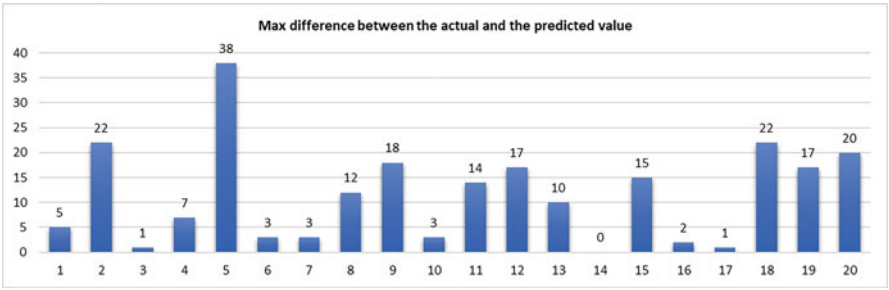


Fig. 7.9 Maximum difference between the actual and predicted vehicles flow values (phase 2)

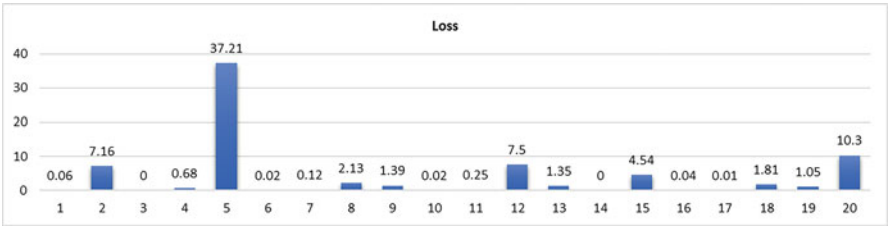


Fig. 7.10 Loss values when predicting vehicles flow in phase 2

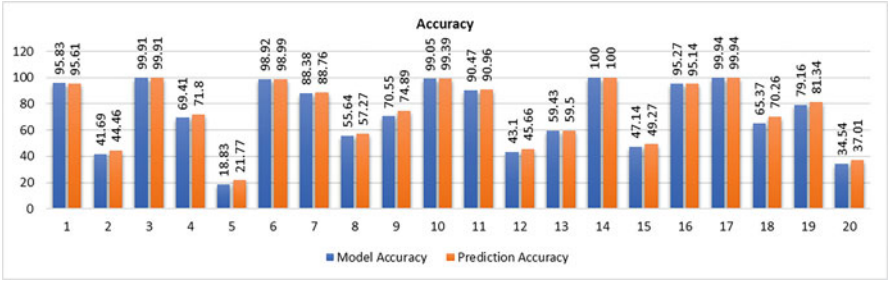


Fig. 7.11 Comparison of testing and prediction accuracy in phase 2

difference values, we have calculated system generated loss values which are shown in Fig. 7.10.

As we are using the different data for testing and prediction dataset, we have calculated the accuracy for both, testing and prediction processes for all 20 executions of our model in phase 2. Testing accuracy in this case has been generated by the system but the prediction accuracy has been calculated manually by comparing the actual and the predicted vehicles flow values. This shows that our model produced accurate results for both, testing and prediction data subsets. Comparison of testing and prediction accuracy values in phase 2 is shown in Fig. 7.11. In this figure, model accuracy represents the accuracy values obtained during the testing process using testing data subset.

7.7 Conclusion and Future Work

In this work we have used deep learning approach to manage traffic flow in smart cities for disaster management. Deep learning requires a large amount of data for training purpose that could easily be accessed from the traffic departments in smart cities. In this work we have used historic traffic data to predict the traffic flow and its behavior in disaster. The results show very high accuracy rate because of the high correlation between the input data and the output values. The results may differ when same deep learning model is applied on a different type of data. We have plotted MAE and MAPE results for all 20 executions of our model with the same specification. The results show that a specific accuracy rate was maintained in all 20 executions of our model and thus we can say that its output is consistent to a certain extent. In addition to error rates, we have plotted the original and predicted flow values to visualize the difference between the graph trends followed by actual and predicted values graphs. Graphs also show similar trends and prove that there are not big differences between the actual and the predicted values. As mentioned earlier, we mainly have focused in this paper on providing details of the deep learning based traffic prediction approach. Details of the overall evacuation method can be found in our earlier work [12, 13, 22, 35].

Although we have shown excellent results in this work, but this is not guaranteed while working with other traffic data with same or other deep learning models. This could be the result of high uniformity in input data that was used for training and testing purposes, and therefore, the same performance of deep model could not be guaranteed for other datasets. Therefore, we aim to work on different data with many other features including incidents data, etc., to see its impact. This may also help us in predicting the people and other stakeholders behavior in emergency situations and we may model them collectively to present a model to not only manage traffic by flow values but also by including other important factors in that environment as well. We can also use real-time traffic and other data to present an effective traffic management plan in the affected areas and can also use big data technologies to deal with real-time data.

Acknowledgements The authors acknowledge with thanks the technical and financial support from the Deanship of Scientific Research (DSR) at the King Abdulaziz University (KAU), Jeddah, Saudi Arabia, under the grant number G-673-793-38. The work carried out in this paper is supported by the High Performance Computing Center at the King Abdulaziz University, Jeddah.

References

1. Tawalbeh, L., Basalamah, A., Mehmood, R., Tawalbeh, H.: Greener and smarter phones for future cities: characterizing the impact of gps signal strength on power consumption. IEEE Access **4**, 858–868 (2016)
2. Mehmood, R., Alam, F., Albogami, N.N., Katib, I., Albeshri, A., Altowaijri, S.M.: UTiLearn: a personalised ubiquitous teaching and learning system for smart societies. IEEE Access **5**, 2615–2635 (2017)

3. Muhammed, T., Mehmood, R., Albeshri, A., Katib, I.: UbeHealth: a personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities. IEEE Access **6**, 32258–32285 (2018)
4. Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.): Smart Societies, Infrastructure, Technologies and Applications, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST). Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 224. Springer, Cham (2018)
5. Gharaibeh, A., Salahuddin, M.A., Hussini, S.J., Khreishah, A., Khalil, I., Guizani, M., Al-Fuqaha, A.: Smart cities: a survey on data management, security, and enabling technologies. IEEE Commun. Surv. Tutorials **19**(4), 2456–2501 (2017)
6. Su, K., Li, J., Fu, H.: Smart city and the applications. In: 2011 International Conference on Electronics, Communications and Control (ICECC), pp. 1028–1031. IEEE, Piscataway (2011)
7. Mehmood, R., Faisal, M.A., Altowaijri, S.: Future networked healthcare systems: a review and case study. In: Boucadair, M., Jacquenet, C. (eds.): Handbook of Research on Redesigning the Future of Internet Architectures, pp. 531–558. IGI Global, Hershey (2015)
8. Usman, S., Mehmood, R., Katib, I.: Big data and HPC convergence: the cutting edge and outlook. In: International Conference on Smart Cities, Infrastructure, Technologies and Applications (SCITA 2017); Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, vol. 224, pp. 11–26. Springer, Cham (Nov 2018)
9. BBC: Japan Flood: At Least 179 Dead After Worst Weather in Decades (2018)
10. CNN: Japan Floods: Death Toll Rises to 195 as Abe Visits Affected Areas (2018)
11. Nikkei Asian Review: Japan Faces $2bn Price Tag for Flood Rebuilding. https://asia.nikkei.com/Politics/Japan-faces-2bn-price-tag-for-flood-rebuilding (2018). Accessed 20 Nov 2018
12. Alazawi, Z., Alani, O., Abdljabar, M.B., Altowaijri, S., Mehmood, R.: A smart disaster management system for future cities. In: Proceedings of the 2014 ACM International Workshop on Wireless and Mobile Technologies for Smart Cities, pp. 1–10. ACM, New York (2014)
13. Alazawi, Z., Altowaijri, S., Mehmood, R., Abdljabar, M.B.: Intelligent disaster management system based on cloud-enabled vehicular networks. In: 2011 11th International Conference on ITS Telecommunications (ITST), pp. 361–368. IEEE, Piscataway (2011)
14. Büscher, M., Coulton, P., Efstratiou, C., Gellersen, H., Hemment, D., Mehmood, R., Sangiorgi, D.: Intelligent mobility systems: some socio-technical challenges and opportunities. In: International Conference on Communications Infrastructure. Systems and Applications in Europe, pp. 140–152. Springer, Berlin (2009)
15. Mehmood, R., Meriton, R., Graham, G., Hennelly, P., Kumar, M.: Exploring the influence of big data on city transport operations: a Markovian approach. Int. J. Oper. Prod. Manag. **37**(1), 75–104 (2017)
16. Suma, S., Mehmood, R., Albugami, N., Katib, I., Albeshri, A.: Enabling next generation logistics and planning for smarter societies. Proc. Comput. Sci. **109**, 1122–1127 (2017)
17. Suma, S., Mehmood, R., Albeshri, A.: Automatic event detection in smart cities using big data analytics. In: International Conference on Smart Cities, Infrastructure, Technologies and Applications, pp. 111–122. Springer, Berlin (2017)
18. Alomari, E., Mehmood, R.: Analysis of tweets in Arabic language for detection of road traffic conditions. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, vol. 224, pp. 98–110. Springer, Cham (Nov 2018)
19. Mehmood, R., Graham, G.: Big data logistics: a health-care transport capacity sharing model. In: Procedia Computer Science, vol. 64, pp. 1107–1114. Elsevier, Amsterdam (2015)
20. Arfat, Y., Mehmood, R., Albeshri, A.: Parallel shortest path graph computations of United States road network data on apache spark. In: International Conference on Smart Cities, Infrastructure, Technologies and Applications, pp. 323–336. Springer, Berlin (2017)
21. Mehmood, R., Lu, J.A.: Computational Markovian analysis of large systems. J. Manuf. Technol. Manag. **22**(6), 804–817 (2011)

22. Aqib, M., Mehmood, R., Albeshri, A., Alzahrani, A.: Disaster management in smart cities by forecasting traffic plan using deep learning and GPUs. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.): International Conference on Smart Cities, Infrastructure, Technologies and Applications (SCITA 2017): Smart Societies, Infrastructure, Technologies and Applications, pp. 139–154. Springer, Cham (2018)
23. Mehmood, R., Nekovee, M.: Vehicular Ad hoc and grid networks: discussion, design and evaluation. In: Proceedings of The 14th World Congress On Intelligent Transport Systems (ITS), Beijing, October 2007. ITS America, Washington (2007)
24. Gillani, S., Shahzad, F., Qayyum, A., Mehmood, R.: A survey on security in vehicular Ad hoc networks. In: International Workshop on Communication Technologies for Vehicles, pp. 59–74. Springer, Berlin (2013)
25. Alvi, A., Greaves, D., Mehmood, R.: Intra-vehicular verification and control: a two-pronged approach. In: 2010 7th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP 2010), pp. 401–405. IEEE, Piscataway (2010)
26. Nabi, Z., Alvi, A., Mehmood, R.: Towards standardization of in-car sensors. In: International Workshop on Communication Technologies for Vehicles, pp. 216–223. Springer, Berlin (2011)
27. Ayres, G., Mehmood, R.: On discovering road traffic information using virtual reality simulations. In: 11th International Conference on Computer Modelling and Simulation, 2009. UKSIM'09, pp. 411–416. IEEE, Piscataway (2009)
28. Mehmood, R.: Towards understanding intercity traffic interdependencies. In: Proceedings of The 14th World Congress On Intelligent Transport Systems (ITS), Beijing, October 2007. ITS America, Washington (2007)
29. Graham, G., Mehmood, R., Coles, E.: Exploring future cityscapes through urban logistics prototyping: a technical viewpoint. Supply Chain Manag. An Int. J. **20**(3), 341–352 (2015)
30. Arfat, Y., Aqib, M., Mehmood, R., Albeshri, A., Katib, I., Albogami, N., Alzahrani, A.: Enabling smarter societies through mobile big data fogs and clouds. Procedia Comput. Sci. **109**, 1128–1133 (2017)
31. Schlingensiepen, J., Nemtanu, F., Mehmood, R., McCluskey, L.: Autonomic transport management systems-enabler for smart cities, personalized medicine, participation and industry grid/industry 4.0. In: Intelligent Transportation Systems–Problems and Perspectives, pp. 3–35. Springer, Berlin (2016)
32. Schlingensiepen, J., Mehmood, R., Nemtanu, F.C., Niculescu, M.: Increasing sustainability of road transport in European cities and metropolitan areas by facilitating autonomic road transport systems (ARTS). In: Wellnitz, J., Subic, A., Trufin, R. (eds.): Sustainable Automotive Technologies 2013, pp. 201–210. Springer, Ingolstadt (2014)
33. Schlingensiepen, J., Mehmood, R., Nemtanu, F.C.: Framework for an autonomic transport system in smart cities. Cybern. Inf. Technol. **15**(5), 50–62 (2015)
34. Alam, F., Mehmood, R., Katib, I.: D2TFRS: an object recognition method for autonomous vehicles based on RGB and spatial values of pixels. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, vol. 224, pp. 155–168. Springer, Cham (Nov 2018)
35. Alazawi, Z., Abdljabar, M.B., Altowaijri, S., Vegni, A.M., Mehmood, R.: ICDMS: an intelligent cloud based disaster management system for vehicular networks. In: International Workshop on Communication Technologies for Vehicles, pp. 40–56. Springer, Berlin (2012)
36. Alazawi, Z., Alani, O., Abdljabar, M.B., Mehmood, R.: An intelligent disaster management system based evacuation strategies. In: 2014 9th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP), pp. 673–678. IEEE, Piscataway (2014)
37. Alazawi, Z., Alani, O., Abdljabar, M.B., Mehmood, R.: Transportation evacuation strategies based on VANET disaster management system. Procedia Econ. Financ. **18**, 352–360 (Jan 2014)
38. Alam, F., Mehmood, R., Katib, I., Albogami, N.N., Albeshri, A.: Data fusion and IoT for smart ubiquitous environments: a survey. IEEE Access **5**, 9533–9554 (2017)
39. Alam, F., Mehmood, R., Katib, I., Albeshri, A.: Analysis of eight data mining algorithms for smarter internet of things (IoT). Procedia Comput. Sci. **98**, 437–442 (2016)

40. Tsirogiannis, D., Harizopoulos, S., Shah, M.A.: Analyzing the energy efficiency of a database server. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pp. 231–242. ACM, New York (2010)
41. Valentini, G.L., Lassonde, W., Khan, S.U., Min-Allah, N., Madani, S.A., Li, J., Zhang, L., Wang, L., Ghani, N., Kolodziej, J., et al.: An overview of energy efficiency techniques in cluster computing systems. Clust. Comput. **16**(1), 3–15 (2013)
42. Chen, J., Chiang, R.C., Huang, H.H., Venkataramani, G.: Energy-aware writes to non-volatile main memory. In: Proceedings of the 4th Workshop on Power-Aware Computing and Systems, p. 6. ACM, New York (2011)
43. Dhiman, G., Ayoub, R., Rosing, T.: PDRAM: a hybrid pram and dram main memory system. In: DAC'09. 46th ACM/IEEE Design Automation Conference, 2009, pp. 664–669. IEEE, Piscataway (2009)
44. Seshadri, V., Kim, Y., Fallin, C., Lee, D., Ausavarungnirun, R., Pekhimenko, G., Luo, Y., Mutlu, O., Gibbons, P.B., Kozuch, M.A., et al.: RowClone: fast and energy-efficient in-dram bulk data copy and initialization. In: Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 185–197. ACM, New York (2013)
45. Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., Lew, M.S.: Deep learning for visual understanding: a review. Neurocomputing **187**, 27–48 (2016)
46. Ciresan, D.C., Meier, U., Masci, J., Maria Gambardella, L., Schmidhuber, J.: Flexible, high performance convolutional neural networks for image classification. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence, vol. 22, p. 1237, Barcelona, Spain. AAAI Press, Palo Alto (2011)
47. Lighthill, M.J., Whitham, G.B.: On kinematic waves II. A theory of traffic flow on long crowded roads. Proc. R. Soc. Lond. A **229**(1178), 317–345 (1955)
48. Richards, P.I.: Shock waves on the highway. Oper. Res. **4**(1), 42–51 (1956)
49. Greenshields, B., Channing, W., Miller, H., et al.: A study of traffic capacity. In: Highway Research Board Proceedings, vol. 1935. National Research Council (USA), Highway Research Board (1935)
50. Yang, Q., Koutsopoulos, H.N.: A microscopic traffic simulator for evaluation of dynamic traffic management systems. Trans. Res. C Emerg. Technol. **4**(3), 113–130 (1996)
51. Ahmed, K.I.: Modeling Drivers' Acceleration and Lane Changing Behavior. PhD thesis, Massachusetts Institute of Technology, Massachusetts (1999)
52. Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.: Recent development and applications of SUMO—Simulation of Urban MObility. Int. J. Adv. Syst. Meas. **5**(3&4), 128–138 (December 2012)
53. Zhao, Z., Chen, W., Wu, X., Chen, P.C., Liu, J.: LSTM network: a deep learning approach for short-term traffic forecast. IET Intell. Transp. Syst. **11**(2), 68–75 (2017)
54. Yu, R., Li, Y., Shahabi, C., Demiryurek, U., Liu, Y.: Deep learning: a generic approach for extreme condition traffic forecasting. In: Proceedings of the 2017 SIAM International Conference on Data Mining, pp. 777–785. SIAM, Philadelphia (2017)
55. Jia, Y., Wu, J., Du, Y.: Traffic speed prediction using deep learning method. In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pp. 1217–1222. IEEE, Piscataway (2016)
56. Djahel, S., Salehie, M., Tal, I., Jamshidi, P.: Adaptive traffic management for secure and efficient emergency services in smart cities. In: 2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 340–343. IEEE, Piscataway (2013)
57. Dandala, T.T., Krishnamurthy, V., Alwan, R.: Internet of vehicles (IoV) for traffic management. In: International Conference on Computer, Communication and Signal Processing (ICCCSP), 2017, pp. 1–4. IEEE, Piscataway (2017)
58. Rizwan, P., Suresh, K., Babu, M.R.: Real-time smart traffic management system for smart cities by using internet of things and big data. In: International Conference on Emerging Technological Trends (ICETT), pp. 1–7. IEEE, Piscataway (2016)

59. Zhu, F., Li, Z., Chen, S., Xiong, G.: Parallel transportation management and control system and its applications in building smart cities. IEEE Trans. Intell. Transp. Syst. **17**(6), 1576–1585 (2016)
60. Lv, Y., Duan, Y., Kang, W., Li, Z., Wang, F.Y.: Traffic flow prediction with big data: a deep learning approach. IEEE Trans. Intell. Transp. Syst. **16**(2), 865–873 (2015)
61. Berkeley, U.: Caltrans (2005) freeway performance measurement system (PeMS) 5.4. pems.eecs.berkeley.edu/Public, Accessed 30 June 2005
62. Polson, N.G., Sokolov, V.O.: Deep learning for short-term traffic flow prediction. Transp. Res. C Emerg. Technol. **79**, 1–17 (2017)
63. Polson, N., Sokolov, V.: Deep learning predictors for traffic flows. arXiv preprint arXiv:1604.04527 (2016)
64. Ma, X., Yu, H., Wang, Y., Wang, Y.: Large-scale transportation network congestion evolution prediction using deep learning theory. PloS One **10**(3), e0119044 (2015)
65. Chollet, F., et al.: Keras (2015). https://keras.io
66. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015) Software available from https://www.tensorflow.org/

Chapter 8
Parallel Shortest Path Big Data Graph Computations of US Road Network Using Apache Spark: Survey, Architecture, and Evaluation

Yasir Arfat, Sugimiyanto Suma, Rashid Mehmood, and Aiiad Albeshri

8.1 Introduction

Smart applications and infrastructures are increasingly relying on graph computations. We are witnessing a continuous increase in the use of graphs to model real-world problems [1]. The emergence of many graph-based software, programming languages, graph databases, and benchmarks—such as ArangoDB, Neo4j, Sparksee, Gremlin, and Graph 500—provide the evidence for the increasing popularity of graph-based computing. Graph analytics plays an important role in information discovery and problem solving. A graph can be any real-life application that can be used to find a relation, route, or a path. Graphs have many applications such as image analysis [2], social network analysis [3, 4], smart cities [5–7], communication networks [8–14], scientific and high performance computing [15–20], transportation systems [21], Web analyses [22], healthcare [23–25], and biological analyses [26]. In these applications, a large amount of data is being generated every second, commonly referred to as big data.

Big Data refers to the "emerging technologies that are designed to extract value from data having four V's characteristics; volume, variety, velocity and veracity" [27, 28]. Volume defines the generation and collection of the vast amount of data.

Y. Arfat · A. Albeshri
Department of Computer Science, FCIT, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: yqasim@stu.kau.edu.sa; aaalbeshri@kau.edu.sa

S. Suma
Division of Data, Department of Engineering, Kumparan, Jakarta Selatan, Indonesia
e-mail: sugimiyanto.sugimiyanto@kumparan.com

R. Mehmood (✉)
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: RMehmood@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_8

Variety defines the type of the data stored or generated. Types include structured, semi-structured, and unstructured data. Velocity describes the timeline related to the generation and processing of big data. Veracity refers to the challenges related to the uncertainty in data. Big Data V's and Graphs have a close relationship. For example, volume could represent the number of edges and nodes, and velocity could be considered as the graph's streaming edges. A graph could be uncertain (veracity) and has the variety characteristics because data sources could vary.

The processing of graphs in a distributed environment is a great challenge due to the size of the graph. Typically, a large graph is partitioned for processing. A graph can be partitioned to balance the load on the various machines in a cluster. These partitions are processed in a parallel distributed environment. For the computation of the graph data on the distributed platform, there is a need for scalability and efficiency. These are the two key elements to achieve good performance. We also need to move our data closer to computation to minimize the overhead of data transfer among the nodes in the cluster. Load balancing and data locality plays a major role in achieving this purpose. It can utilize the whole resource of the system during processing. Moreover, as mentioned earlier, big data cannot be processed by traditional tools and technologies. There are many platforms for graph processing, but these platforms have performance issues. Parallel computation of large graphs is a common problem. Therefore, in this scenario parallel distributed platforms are suitable for processing large graphs. In this work, we have used the GraphX [29–31] for parallel distributed graph processing which is a widely used framework for the graph processing. The big data platform that we have used for distributed graph computing of shortest paths is Apache Spark [32].

This chapter extends our earlier work on single source shortest path computations of big data road network graphs using Apache Spark. In our earlier work [33], we had used the US road network data, modelled as graphs, and calculated shortest paths between two vertices over a varying number of up to 368 compute cores. The experiments were performed on the Aziz supercomputer (a former Top500 machine [34]). We had analyzed Spark's parallelization behavior by solving problems of varying graph sizes, i.e., various states of the USA with up to over 23 million vertices and 58 million edges.

We focus in this chapter on computing a set of large varying number of shortest path queries on a (source, destination) vertex pair. The number of queries used are 10, 100, 1 K, 10 K, 100 K, and 1 M queries executed over up to 230 CPU cores. We achieve good performance, and as expected, the speedup is dependent on both the size of the data and the number of parallel nodes. In addition to the extended results, this chapter provides a detailed literature on shortest path graph computations. The system architecture for graph computing in Spark is explained with additional details using the architecture depiction and elaborated algorithms. We call our system, the Big Data Shortest Path Graph Computing (BDSPG) system.

The rest of the chapter is organized as follows. Section 8.2 gives background and literature review. Section 8.3 describes the design and methodology of the BDSPG system. Section 8.4 presents the analysis of result. The conclusions and future directions are given in Sect. 8.5.

8.2 Literature Review

Smart urban infrastructure greatly replies on smart mobility designs. Many approaches have been proposed to address smart mobility-related challenges [35]. These include, among many others, modelling and simulation-based approaches [36, 37], location-based services [38], telematics [39], social media-based approaches [40–42], approaches based on vehicular networks (VANETs) and systems [43–45], autonomic mobility management [46, 47], autonomous driving [48], mobility in emergency situations [49–54], approaches to improve urban logistics [40, 55], and big data-based approaches [40–42, 56]. A recent book discusses several smart society proposals on infrastructure and applications including smart mobility [7]. Many mobility problems naturally map to graph-based computations; shortest path computations are one of them and are of great significance in smart mobility infrastructure designs. In this section, we discuss state-of-the-art work from the literature on graph-based road network shortest path computations.

Quddus and Washington developed an algorithm to find the shortest path between two points called weight-based shortest path and vehicle trajectory aided map-matching (stMM) [57]. It improves the map-matching of low-frequency positioning data on a roadmap. They exploit a well-known A* search algorithm. They tested the performance of proposed approach with collected data from rural, suburban, and urban areas in Nottingham and Birmingham, UK. Szucs designed and implemented a model and an algorithm for route planning in road network [58]. They proposed a solution that also aims to find the equilibrium in the path optimization problem. The proposed approach takes the uncertainty of state information of roads, their uncertainty and influencing factors into account. The system is based on the Dempster-Shafer theory, which helps to model the uncertainty and Dijkstra's algorithm which allows finding the best route. Feng et al. proposed an improvement of alternative route calculation, based on alternatives figures [59]. They exploit a bidirectional Dijkstra algorithm to explore the route. They introduced three quotas to measure the quality of an Alternative Figures (AG). They introduce the concept of pheromones into the Plateau method and enhance the ability of Plateau method to find a meaningful alternative road.

Zeng and Church demonstrated the relative value of A* algorithm to solve simple point-to-point shortest path problems on real road networks [60]. It is applied to road networks from two counties of California, USA. They state that Dijkstra algorithm can be improved by taking advantage of network properties associated with GIS-source data. Whereupon, Dijkstra does not take advantage of the spatial attributes which are available in a GIS setting, while A* can take the advantage of spatial coordinates in trimming the search to find the shortest path. Malewicz et al. proposed Pregel, a framework for large-scale graph processing [61]. The framework is similar in concept to MapReduce. It provides users with a natural API for programming graph algorithms while managing the details of distribution invisibly, including messaging and fault tolerance. It contributes providing a suitable system for large-

scale graph computing. They deployed dozens of Pregel applications. The users report that the API is intuitive, easy to use, and flexible. The experiment shows that the performance, scalability, and fault tolerance of proposed framework are satisfactory for computing graph jobs with billions of vertices.

Yan et al. proposed a framework called Graphine for graph-parallel computation in multicore clusters [62]. It addresses the problem of existing distributed graph-parallel frameworks which cannot scale well with the increasing number of cores per node. They implemented the proposed framework and evaluated it. The experiment result shows that their proposed framework achieves sublinear scalability with the number of nodes, a number of cores per node, and graph size up to one billion vertices, as well as achieves 2∼15 times faster than the state-of-the-art Power Graph on a cluster with 16 multicore nodes. Selim and Zhan proposed an algorithm and data reduction technique based on data nodes in large networks dataset [63]. It is done by computing similarity computation, maximum similarity clique (MSC), and then finding the shortest path due to the data reduction in the graph. The technique aims to reduce the network that will have a significant impact regarding performance (shortest time and faster analysis) on calculating the shortest path. The proposed technique takes into account shortest path problem between two nodes in a large undirected network graph. The result shows that their proposed technique beats up Dijkstra's shortest path algorithm with large datasets with respect to execution time. Zhou et al. presented a new graph processing framework based on Google's Pregel called P++ [64]. The proposed framework aims to reduce the system overhead for algorithms that require many iterations in Pregel. It extends Pregel by some new terms such as introducing a new data structure, internal compute, super-vertex, and new API. Their proposed approach has been evaluated by using real datasets with cases Shortest Path and PageRank. The result shows that their proposed technique demonstrate its superior performance.

Cao et al. proposed an approach for solving the stochastic shortest path problem in vehicle routing [65]. It aims to find the optimal path that maximizes the probability of arriving at specified destination before the given deadline. Their approach is data-driven which explores big data generated in traffic. They evaluated the performance using a real traffic data extracted from real GPS trajectories of vehicles in road network of Munich city, which consists of 170 nodes and 277 edges. The experiment result shows that the proposed approach outperforms traditional methods. Hou U et al. developed a framework to solve online shortest path problem called live traffic index (LTI) [66]. The proposed framework aims for computing the shortest path according to live traffic conditions. It enables drivers to effectively and quickly get the live traffic information on the broadcasting channel. There is no existing efficient solution that can offer affordable costs for online shortest path computation at both client and server sides. The conventional architecture scales poorly with the number of clients. Their approach is that the server collects live traffic information and distributes it over radio or wireless network. They evaluated their approach with four different road maps, including New York City, San Francisco bay area road map, San Joaquin road map, and Oldenburg road map.

The result shows that their proposed method reach optimal solution in terms of four performance factors for online path computation.

Strehler et al. developed a model called fully polynomial-time approximation scheme (FPTAS) for finding shortest energy-efficient routes for electric and hybrid vehicles [67]. It aims to resolve the problem of electric and hybrid vehicles regarding the shortest path problem and planning of the trip, whereupon recharging an electric car takes longer than refilling fossil fuels car. Their contribution is introducing a general model for the routing of hybrid and electric vehicles with intermediate stops at charging station and convertible resources. They are using Matlab to represent and test their model. The used dataset are engine model, topographical information, and road data of German. They are in improvement phase that the running time of the proposed algorithms may not be suitable for practical purposes, particularly, when it is running on a mobile phone or on an in-car device. Hong et al. developed a multicore computing approach to find shortest route from single source and single destination while avoiding obstacles [68]. Whereupon, the existing approaches have limited ability in dealing with real-time analysis in big data environments. They use multicore computing to speed up the computation and analysis using Python's official Multiprocessing library. Thus, the parallelization is core based. The approach itself exploits the notion of a convex hull for evaluating obstacles and constructing pathways iteratively. The experiment result shows their proposed approach for parallel processing has significant improvements over sequential computing for wayfinding and navigation tasks with a large number of obstacles in complex urban area. Mozes et al. developed an algorithm by combining two techniques for computing shortest paths in directed planar graphs [69]. The two combined techniques are STOC'94 and FOCS'01. It aims to remove the log n dependency of the shortest path algorithm in the running time, in order to have better and optimal performance. The theoretical proving shows that their proposed technique obtains a speedup over previous algorithms for solving shortest-path problem.

In this work, Abraham et al. [70] have worked on the point to point the shortest path computations on the road network data. They modelled the road network as a graph having highways with low dimension. The algorithm they named Hub labels for computation of shortest path. The authors claim that it works faster for all types of queries. However, they have not used the parallel implementation of an algorithm. The performance this might suffer from significant data computation of this algorithm. In this chapter, they have not used the US road network dataset. It uses a general algorithm for the computation of road network graph data. Sanders et al. [71] have presented the real-world road network processing algorithms. They claim that algorithm takes less as compared to the Dijkstra. In this work, they have not used the parallel implementation of the algorithm. They also did not use the big data computation. Peng et al. [72] has presented a framework for the computation of the road network distance using a single source-target pair. In presented algorithms, they mapped the distance into a distributed structure of hash. For the implementation, they used the Apache Spark and in memory computation for the distance of road network computation. They experimented their algorithm

using US and NYC road network dataset. Zhu et al. [73] have proposed an index structure called Arterial Hierarchy (AH) for the shortest and distance queries in a road network. They argue that existing work concentrates on the practical or asymptotic performance. The problem with state of the art was worst regarding space and time. The primary objective of this chapter was to minimize the gap between theory and practice for shortest path queries on road network. For the evaluation, they have used the 20 million nodes. The proposed technique performs better than existing approaches for road network dataset. Moreover, in this work, they have not used the weighted road network graph data.

Zheng et al. [74] have presented all pair shortest path algorithm. The proposed algorithm was an alternative to the Floyd-Warshall. They implemented their algorithm using Apache Spark and analyzed the performance of their algorithm. They argue that the performance of Floyd-Warshall algorithm suffered using Apache Spark due to a large number of global updates. To solve this issue, they have used the fewer global update steps based on computation that has been done on each iteration. As a result, they showed that their algorithm performs better than Floyd-Warshall algorithm. However, their work is different as compared to our work. We are parallelizing the shortest path between two vertices source and target. Djidjev et al. [75] have presented all pair shortest path algorithm using GPU cluster. They have used both centralized and decentralized computation for the all pair shortest path algorithm. They have presented the two algorithms that use the Floyd-Warshall method. For implementation, they have used the multi-GPU cluster. They have also used the California state road network dataset that consists of 1.9 million vertices and five million edges. Aridhi et al. [76] have presented the shortest path algorithm on the base of MapReduce. To solve the shortest path problem in an efficient way, they have partitioned the graph into subgraphs then they process it parallel. The algorithm they have proposed is an iterative whose performance will suffer when these are large of input data due to its iterative nature. For an experiment, they have used the French road network dataset from the OpenStreetMap. For the computation, they have used the Hadoop and MapReduce. Faro et al. [77] have presented a shortest path all pair algorithm with and without traffic congestions on the road network. The main objective of this chapter was to find the fastest shortest path on road network. They implemented the proposed all pair shortest algorithm parallel. First, they tried to find the shortest path then tried to find the alternate shortest path in case of traffic congestion. They implemented their algorithm using the GPU. They have not used any road network dataset, neither Spark nor Hadoop.

Kajdanowicz et al. [78] used the Bulk Synchronous Parallel (BSP), map-side join, and MapReduce for the graph computation. They applied these approaches for the single source shortest path (SSSP) and relational influence propagation (RIP) for collective classification of graph vertices. They stated that using BSP iterative graph processing perform better as compared to MapReduce. Liu et al. [79] have proposed a framework for parallel processing of large graph to solve the issue of communication between partitions, unbalanced partition, and replication of vertices. This framework uses three different greedy graph partitioning algorithms. They run these algorithms using the various dataset and observed that whether

these algorithms can solve the issues of graph partitioning based on the specific needs. The major objective of this framework was to balance the load and reduce the bandwidth. Wang et al. [80] proposed a technique for k-plex enumeration and maximal clique approach. Using the binary graph partitioning approach, find the dense subgraph from the graph. It parallel process each partition of the graph by dividing the graph. MapReduce was used for implementation. Braun et al. [81] presented a new approach for social network analysis for knowledge-based systems. The major objective of this technique is to mine the interests of social network and represent as graph. The directed graph has been used for relationship analysis and undirected graph has been used to capture mutual friends. They have used the Facebook and Twitter dataset to analyze the performance of the proposed approach.

Laboshin et al. [82] proposed a new framework based on MapReduce to analyze the web traffic. The major objective of proposed framework was to scale the storage and computing resources for the extensive network. Liu et al. [83] proposed a clustering algorithm for the distributed density. This algorithm solves the issues in distance-based algorithms. This algorithm calculates the distance among all pairs of vertices. The authors claim that using this algorithm computational cost will be reduced. They implemented their algorithm using Apache GraphX [29, 30]. Aridhi et al. [84] investigated different frameworks for mining of big graph. The major focus was to use the mining algorithm for pattern mining that consists of the discovering useful information from the huge graph dataset. They analyzed comprehensively different mining techniques for the large graphs. Drosou et al. [85] proposed an enhanced Graph Analytical Platform (GAP) framework for processing of large graph dataset. This framework uses the top-down approach for mining of huge graph dataset. It provides the strength to features like HR clustering. It is an effective framework for the big data getting useful insights.

Zhao et al. [86] evaluated various graph computation platforms. They did comparison between graph- and data-parallel platform for processing of large dataset. They found out that graph-parallel platforms perform better for resource utilization and graph computation as compared to data-parallel platforms. However, data-parallel platform for graph processing is superior in performance regarding size. Mohan [87] et al. compared the graph computation platforms for large data processing using the key features and performance. Miller et al. [88] investigated the graph analytics from perspective of query processing. There are issues in finding the interesting information from the graph whether it's a shortest path or pattern matching from the graph. They also introduced algorithms which show that vertex centric and graph centric algorithms are easily parallelizable. They stated that MapReduce is not an ideal platform for the iterative algorithm.

Chakaravarthy et al. [89] proposed an algorithm that is derived from the Delta-stepping and Bellman-Ford algorithms. The primary objective was to categorize the edges, minimize the traffic of inner vertices, and optimize the directions. They applied the single source shortest path (SSSP) to get the shortest path between the vertices. Yinglong et al. [90] stated that big data analytics are essential for the entities that can be represented as graph. It is the main challenge for the computation of graph bases patterns. They presented a new architecture that allow

users to organize the data for parallel computation. This architecture has three components: graph storage, analytics, and visualization. They evaluated the data locality for the processing and effects on the performance of cache memory on a processor. Zhang et al. [91] presented an algorithm for the fast graph search. This algorithm converts the completed graphs into vectorial representations on the basis of prototype in the database. So, it accelerates the query efficiency in a Euclidean space by using locality-sensitive hashing. They examined their proposed approach using real dataset that gets higher performance regarding accuracy and efficiency. Pollard et al. [92] proposed a new technique for the parallel graph processing platforms analysis based on the performance and scalability. They used the breadth first search, page rank, and single source shortest for the analysis of power consumption and performance with packages of graph processing with various datasets.

Table 8.1 provides a comparison of various shortest path graph computation approaches. The table includes information for each work regarding aim and objectives, the approach used, the dataset sources, the type of datasets, the platforms used, research gap, and comments. We would have preferred to include the names of authors of the respective works in a separate column in the table but these were omitted to save space and fit the table in as few pages as possible.

8.3 Methodology and Design

This section details the methodology and design of our Big Data Shortest Path Graph Computing (BDSPG) System.

Figure 8.1 shows the architecture for shortest path computations. First, it will take the graph data as input for the computation of shortest path. This data can be directed or undirected graph data but in our work, we are using undirected graph dataset. Once we have data we have uploaded any distributed file system, so nodes in the cluster can easily access this data. There can be any distributed file system such as FEFS, NEFS, and HDFS. But in our work, we are using HDFS. After keeping input graph data, we build the graph and perform the one pair shortest path (OPSP) using GraphX [31]. After computation of OPSP, we shall get the shortest path having total distance and vertices in the source and target vertex.

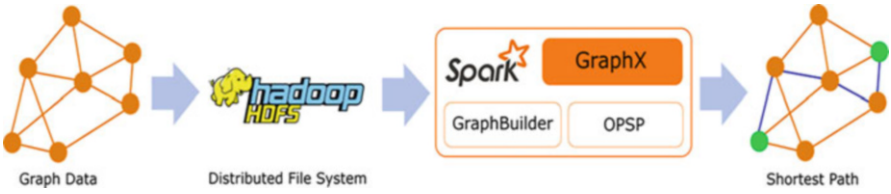


Fig. 8.1 The Big Data Shortest Path Graph Computing (BDSPG) System Architecture

Table 8.1 Comparison of various shortest path graph computation approaches

Aim and objectives [Source]	Approach	Dataset	Dataset type	Platforms	Research gap gap/comment
Shortest path between two points [57]	Find the shortest path between two points called weight-based shortest path	Collected data from rural, suburban, and urban areas in Nottingham and Birmingham, UK	Road network	Not clear	Big data platform, e.g., apache spark not used
Find the equilibrium in the path optimization problem [58]	An algorithm for route planning in road network	Transport network Vertices: 4111 Edges: 5443	Road network	C programming language	Spark not used
Enhance the ability of plateau method to find a meaningful alternative road [59]	Bidirectional Dijkstra algorithm to explore the route	Beijing China Vertices: 153275 Edges: 433719	Road network	Java	Not distributed, only one node used
Point-to-point shortest path [60]	A* algorithm to solve simple point-to-point shortest path problems on real road networks	Road network data from two counties in California. Santa Barbara: Vertices 33,074, edges: 78144, Los Angeles: Vertices: 195233, edges: 532178	Road network	C programming with visual studio	Apache spark is not used
Large SSSP graph processing [61]	A natural API for programming graph algorithms while managing the details of distribution invisibly	Claims to use a framework for billions of edges and vertices	Any type of graph data	C++	

(continued)

Table 8.1 (continued)

Aim and objectives [Source]	Approach	Dataset	Dataset type	Platforms	Research gap gap/comment
To solve problem of existing distributed graph-parallel frameworks which cannot scale well with the increasing number of cores per node [62]	Graphine for graph-parallel computation in multicore clusters	LiveJournal:(5,363,260 vertices, and 79,023,142 edges) Hollywood (2,180,759 vertices and 228,985,632 edges) Arabic:(22,744,080 vertices, and 639,999,458 edges) Twitter: (41,652,230 vertices, and 1,468,365,182 edges. SK: 50,636,154 vertices, and 1,949,412,601 edges. UK: 105,896,555 vertices, and 3,738,733,648 edges	Social media, web, journal	Apache spark, OpenMPI, C	Do not use road network dataset
Reduce the network that will have a significant impact on performance (shortest time and faster analysis) on calculating the shortest path [63]	Computing similarity computation, maximum similarity clique (MSC), and then finding the shortest path due to the data reduction in the graph	XML-based network graph dataset	Network graph		Do not use road network dataset

Aims to reduce the system overhead for algorithms that require many iterations in Pregel [64]	A new graph processing framework based on Google's Pregel called P++. It extends Pregel by some new terms such as introducing a new data structure, internal compute, super-vertex, and new API.	Social network dataset twitter (40 million vertices, > 1 billion edges), and synthetic dataset by using first K vertices and their edges to evaluate the scalability of their proposed framework	Dataset: Social network dataset	P++ framework	They have used the unweighted graph dataset
Find the optimal path that maximizes the probability of arriving at a specified destination before the given deadline [65]	Solving the stochastic shortest path problem in vehicle routing	Road network of Munich city, which consists of 170 nodes and 277 edges. And also weighted graph of travel time, extracted from real GPS trajectories of BMW vehicles	Synthetic and road network dataset	Machine learning	Small dataset, only on single node, not parallel
Compute the shortest path according to the live traffic conditions [66]	A framework to solve online shortest path problem called live traffic index (LTI)	New York City (NYC) (264 k nodes, 733 k edges), San Francisco bay area road map (SF) (174 k nodes, 443 k edges), San Joaquin road map (SJ) (18 k nodes, 48 k edges), and Oldenburg road map (OB) (6 k nodes, 14 k edges)	Road network	Java	Small dataset, not distributed, single node only

(continued)

Table 8.1 (continued)

Aim and objectives [Source]	Approach	Dataset	Dataset type	Platforms	Research gap gap/comment
Resolve the problem of electric and hybrid vehicles regarding the shortest path problem and planning of the trip [67]	Developed a model called fully polynomial-time approximation scheme (FPTAS) for finding shortest energy-efficient routes for electric and hybrid vehicles	Synthetic for simple evaluation. Real dataset topographical information, and road data from Germany. Data size not clear	Synthetic and road network dataset.	They used Matlab to represent and test their model	No distributed implementation, single node only
Remove limitations of existing approaches and limited ability in dealing with real-time analysis in big data environments [68]	Developed a multicore computing approach to find shortest route from single source and single destination while avoiding obstacles	The initial area of focus is the Arizona State University campus in Tempe, Arizona. The campus has 179 buildings that represent obstacles to direct travel between an origin and destination. With total 2688 pairs of origin-destination	Not clear	Sequential and parallel process in Python	Road network dataset not used, small dataset, apache spark not used
Remove the log n dependency of the shortest-path algorithm in the running time, in order to have better and optimal performance [69]	An algorithm by combining two techniques for computing shortest paths in directed planar graphs	Theoretical proof, no dataset	N/A	N/A	Not distributed, only one node

Exact point-to-point shortest paths in road networks [70]	Hub labels (HL), a labeling algorithm to compute shortest path	DIMACS Road network dataset (USA road network and Europe road network)	Road network	C++ and Microsoft visual C++	Not parallel
Find shortest path [71]	Presented a new speedup technique for route planning that exploits the hierarchy inherent in real-world road networks	DIMACS Road network (USA, Europe, Germany)	Road network	C++	No parallel or big data implementation
Speed up complex spatial analytical queries and evaluate a large number of network distance queries which are posed as a large set containing N source-target pairs [72]	Presented a framework for the computation of the road network distance using a single source-target pair	US (V:23 M, E:58 M), Bay Area (V: 758,104,E:1 M) and NYC (V: 264,346, E: 733,846) road network dataset	Road network	Apache spark	
Minimize the gap between theory and practice for shortest path queries on road network [73]	Proposed an index structure called arterial hierarchy (AH) for the shortest and distance queries in a road network	Delaware (48,812 V 120,489E) New Hampshire (115,055 V, 264218E) Maine (187,315 V, 422998 E) Colorado (435,666 V, 1057066E) Florida (1,070,376 V,2712798E) California and Nevada (1,890,815 V, 4657742E) Eastern US 3598623 V, 8778114E) Western US (6,262,104 V, 15248146) Central US (14,081,816 V,34292496E) United States (23 M, 58E)	Road network	C++	

(continued)

Table 8.1 (continued)

Aim and objectives [Source]	Approach	Dataset	Dataset type	Platforms	Research gap gap/comment
Improve the performance APSP using Apache Spark [74]	All-pairs-shortest-paths (APSP)	Synthetic distance matrices	Synthetic dataset	Apache spark	No real dataset
To use both centralized and decentralized computation for the all pair shortest path algorithm [75]	All pair shortest path algorithm using GPU cluster	Graph dataset 1.9 M vertices, 5 M edges	Road network	GPU cluster	
Solve the shortest path problem in an efficient way, partitioned the graph into subgraphs then process it parallel [76]	Presented the shortest path algorithm using MapReduce	French road network dataset from the OpenStreetMap	Road network	Hadoop and MapReduce	MapReduce cannot give good performance under iterative environment
Find the fastest shortest path on road network [77]	Presented a shortest path all pair algorithm with and without traffic congestions on the road network	N/A	N/A	GPU	No road network dataset, not spark or Hadoop
Use BSP iterative graph processing for better performance [78]	The bulk synchronous parallel (BSP), map-side join and MapReduce for the graph computation	Twitter, YouTube, and tele with millions of edges	Social media	MapReduce	MapReduce used
Solve the issues of communication between partitions, unbalanced partition, and replication of vertices [79]	Proposed a framework for parallel processing of large graphs	There are several datasets but largest one is Facebook: 1000001 V, 23728298 E: Live journal: 3997962 V, 34681189 E. (Facebook, twitter, live media)	Real dataset, synthetic dataset, used the SNAP dataset	MapReduce	MapReduce

Effective load balancing and efficient parallel performance [80]	Proposed a technique for k-plex enumeration and maximal clique approach	Social networks, wikivote, epinions, Slashdot0902, Gowalla_edges, youtube Pokec, WikiTalk, web graphs, uk2005 it2004, BerkStan, WebGoogle, WikiComm, wikipedia2009, miscellaneous, networks, HepPh, EuAll, dblp2012 skitter	The largest dataset they have used is: Pokec (1,632,803 V, 22,301,964 E)	MapReduce	No road network dataset
Mine the interests of social network and represent as graph [81]	Presented a new approach for social network analysis for knowledge-based systems	Ego-Facebook (4039 V, and 88,234E) Ego-Twitter (81,306 V, 1,768,149E)	Social media	Apache Hadoop 0.20.0	They have not used weighted graph
Scale the storage and computing resources for the extensive network [82]	A new framework based on MapReduce to analyze the web traffic	N/A	N/A	N/A	A framework to process large dataset
Solve the issues in distance-based algorithms [83]	Proposed a clustering algorithm for distributed density	News dataset contains 10 topics, and 47,956 texts	News dataset	Apache Spark and GraphX [93]	No road network dataset
Use mining algorithm for pattern mining for discovering useful information from a huge graph dataset [84]	Investigated different frameworks for mining of big graph	N/A	N/A	N/A	It is a review of different graph mining frameworks

(continued)

Table 8.1 (continued)

Aim and objectives [Source]	Approach	Dataset	Dataset type	Platforms	Research gap gap/comment
It provides the strength to features like HR clustering [85]	Proposed an enhanced graph analytical platform (GAP) framework for processing of large graph dataset	Twitter dataset and synthetic data mobile-based dataset. Size of this dataset is not given	Social media and synthetic	Not clear	Path graph analysis not used, no clear mention of anything about graph processing platforms
Comparison between graph and data parallel platform for processing of large dataset [86]	Evaluations of various graph computation platforms	Facebook (61,876,615 V, 336,776,269E). LiveJournal (4,847,571 V, 68,993,773E) and DBLP (317,080 V, 1,049,866E)	Social networking dataset	Spark-0.9.0 PowerGraph-2.2 Giraph 1.0.0 GPS-0.0.1	No road network dataset
Comparison using the key features and performance [87]	Compared the graph computation platforms for large data processing	Collaboration (317,080 V, 1,049,866 E) Communication (36,692 V, 183831 E) CA road network (1,965,206 V, 2,766,607 E)	Real graph dataset and synthetics	Hadoop, Giraph, MapReduce, and BSP	Not directly related to our work. However, it shows how to process large graphs using BSP. They used road unweighted graph dataset
Finding interesting information from the graph whether it's a shortest path or pattern matching from the graph [88]	Investigated graph analytics from the perspective of query processing	N/A	N/A	Theoretical evaluations	This work is related to future trends and direction of graph analytics

Categories the edges, minimize the traffic of inner vertices and optimize the direction [89]	Proposed an algorithm that is derived from the delta-stepping and bellman-ford algorithms	R-MAT graph with 2^{38} vertices and 2^{42} edges on 32,768 blue gene/Q nodes. USA road network with 2.7 M vertices and 6.8 M edges. Friendster 63 M vertices and 1.8 B edges. Twitter 41 M vertices and 1.4B edges	Synthetic R-MAT and real-world graphs	GPU and NUMA multicores	Uses SSSP path
Computation of graph-based patterns [90]	Presented a new architecture that allows users to organize the data for parallel computation	10 billion vertices and 200 billion edges	Not clear	PERCH (POWER7+ cluster)	Uses SSSP
Main aim was to accelerate the query efficiency in a Euclidean space by using locality sensitive hashing [91]	Presented an algorithm for fast graph search	NCI1 and NCI109, mutagenicity dataset	The dataset provided related to cancer and chemicals	Empirical evaluations	
Analysis of performance and scalability [92]	Proposed a new technique for parallel graph processing platforms	Cit-patents (3,774,768 vertices and 16,518,948 edges) Dota-league	Synthetic and real-world datasets	Graph-mat, the Graph500, the graph algorithm platform benchmark suite, GraphBIG, and PowerGraph and C	No weighted graph network dataset, have used shortest path computations

We propose an approach for the parallel shortest path computation with multiple queries of a pair of vertices using Apache Spark. In this approach, we have two functions: The One Pair Shortest Path (OPSP) algorithm to find the best route between a pair of vertices, and the main driver program which builds the graph, constructs and parallelizes the queries, and invokes OPSP function. Algorithm 1 (Fig. 8.2) shows the OPSP algorithm. In this algorithm, we employ the concept of the well-known Dijkstra algorithm to find the optimal route between source and destination in a graph problem. This algorithm first explores the neighbor vertices of the current vertex from *distPaths[0]* (the path of minimum distance from *src* to *dest*), inserts the neighbor's vertex id to a set of explored vertices *exp[],* if the neighbor vertices have not been explored in advance, keeps track of explored paths from source to the neighbors (the list of vertices to reach the neighbors) and its distance (*neboursPath.concat(distPathRest)*), picks the path with minimum distance to be explored further (*sortByDist()* ascending), and calls the OPSP function itself (recursive) until the path with minimum distance meets the destination, then will return the minimum distance and the paths to reach the destination (*dist, paths.reverse()*).

Algorithm 2 (Fig. 8.3) shows the main driver program. It builds the graph, the queries, and executes the queries with OPSP algorithm. First, the program builds a graph from the given input of vertices and edges *G(V,E)*. Then, constructs queries *q* from the given input of list of *queries(src,dst)*, which contains multiple pairs of *src* and *dest*. Furthermore, *q* is partitioned with *np* size and becomes *Q(src,dst)*. Afterwards, *Q(src,dst), G(V,E),* and initialization variable *exp[]* as a set of explored vertices, and *distPaths(list(k,v[]))* as an initial step of 0 distance and source vertex are passed to OPSP function in Algorithm 1. Multiple queries of *Q(src,dst)* are executed in parallel by multiple executors in cluster nodes of Spark. Thus, each executor computes different multiple queries at the same time *t*.

8.3.1 Dataset

We have used the DIMACS [94] dataset. The DIMACS is a collection of various datasets. It also has road network dataset containing more than 50 states of the USA and various districts. It is an undirected weighted graph that consists of millions of edges and nodes. We considered in our experiments the entire US dataset. We have also investigated in this chapter results for five different states of the USA. These are District of Columbia (DC), Rhode Island (RI), Colorado (CO), Florida (FL), and California (CA). Each node has node id, latitude, and longitude. Every edge also has source node id, target node id, travel time, distance, and category of road. Table 8.2 shows the number of edges and vertices in different states as well as for the complete US road network. Figure 8.4 graphically displays degree of vertices for selected states and whole US road network.

We also have visualized road network dataset using Gephi [95]. We have only visualized the DC and RI state data set as shown in Figs. 8.5 and 8.6, respectively.

Algorithm 1: One Pair Shortest Path (OPSP)

```
Input  : G(V, E) ← graph data
         Q(src, dest) ← list of queries
         exp[ ] ← init of explored vertices
         distPaths(list(k, v[ ])) ← init of explored paths with distance
Output: shortest path of Q(src, dst)
if distPaths == null then
    return (0, ∅)
else
    foreach i ← distPaths.iterate() do
        minDistPath((k, v[ ])) ← distPaths[0];
        disPathRest ← distPaths[1, distPaths.length()];
        dist ← minDistPath.k;
        paths[ ] ← minDistPath.v;
        head ← paths[0];
        rest ← paths[1, paths.length()];
        if head == Q_i.dst then
            return (dist, paths.reverse())
        else
            neboursPath ← list((0, [ ]));
            nb ← G(head);
            de ← nb.distance;
            ve ← nb.vertice;
            foreach nb.iterate() do
                if exp[ ].contains(ve) then
                    continue
                else
                    neboursPath ← (dist + de, ve.concat(paths))
                end
            end
            combDistPath ← neboursPath.concat(distPathRest);
            sortDistPath ← combDistPath.sortByDist();
            OPSP(G, Q, exp.concat(head), sortDistPath);
        end
    end
end
```

Fig. 8.2 The One Pair Shortest Path (OPSP) Algorithm

We could not visualize the other states data due to the large size which cannot be handled on a single PC. We have only visualized two states to perceive the structure of road network datasets. We will look into visualizing larger datasets using Spark in the future.

Algorithm 2: Main Function

Input : $E \leftarrow$ list of edges
$V \leftarrow$ list of vertices
$Q(src, dest) \leftarrow$ list of queries
$np \leftarrow$ number of partition

Output: file of shortest path list

1 $G(V, E) \leftarrow$ RDD(V,E);
2 $q \leftarrow$ queries(src, dst) ;
3 $Q(src, dst) \leftarrow$ RDD(q).repartition(np);
4 $SPL \leftarrow OPSP((G(V, E),\ Q(src, dst)), exp[\,], distPaths(list(k, v[\,]))$;
5 $SPL.saveAsFile$;

Fig. 8.3 The Master Algorithm

Table 8.2 USA road network dataset

Name of Road Network	Vertices	Edges	Type
District of Columbia (DC)	9559	14,909	Undirected
Rhode Island (RI)	53,658	69,213	Undirected
Colorado (CO)	435,666	1,057,066	Undirected
Florida (FL)	1,070,376	2,712,798	Undirected
California (CA)	1,890,815	4,657,742	Undirected
USA (whole country)	23,947,347	58,333,344	Undirected

8.4 Results and Discussion

For experimental setup, we have built a Spark cluster setup on the Aziz supercomputer [34]. In this configuration, we have used different number of nodes, varying from one to sixteen. We have used Apache Hadoop HDFS to store input and output data. Apache Spark has been used for the data processing. The Master and Salve Spark nodes used on the Aziz supercomputer have the following configuration.

- Linux CentOS, JDK 1.7, Dual Socket Intel Xeon E5-2695v2 12-core processor, 2.4 GHz, Total 24 cores, 96GB RAM, Apache Spark 2.0.1, GraphX Apache Hadoop HDFS.

8.4.1 Single Shortest Path Query Results

In our earlier work [33], we had presented results for a single shortest path query on up to 16 nodes (368 cores) for the USA states DC, RI, CO, FL, CA, and the whole US road network with up to over 23 million vertices and 58 million edges (see Table 8.2). See [33] for the detailed results and analysis.

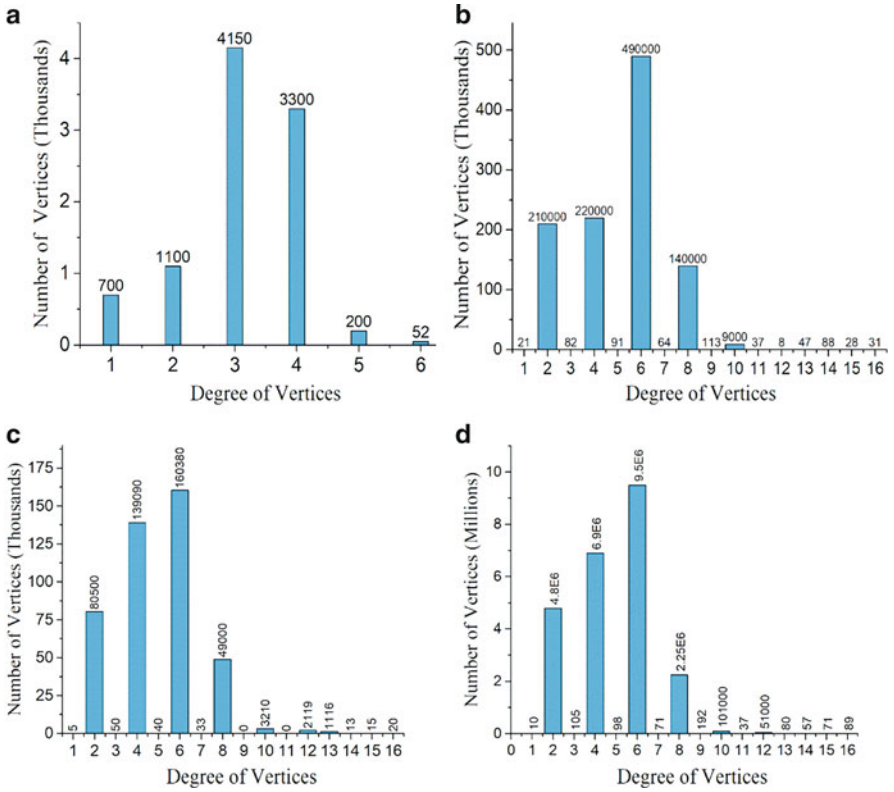


Fig. 8.4 Visualization of (**a**) District of Columbia road network (**b**) Florida road network (**c**) Colorado road network (**d**) Whole US road network

Fig. 8.5 District of Columbia road network

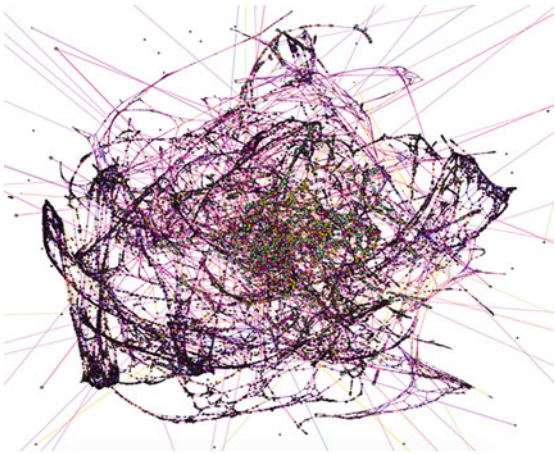


Fig. 8.6 Rhode Island road network

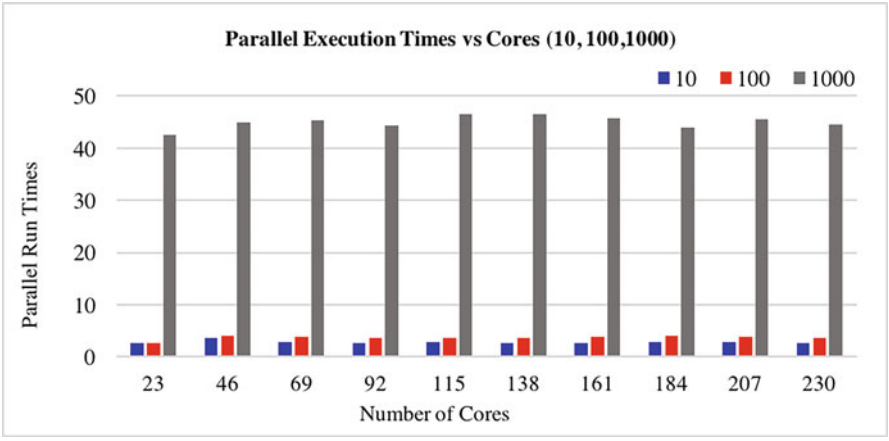


Fig. 8.7 Parallel execution time of varying number of cores

8.4.2 Multiple Shortest Path Query Results

The aim here is to investigate and achieve high performance in finding the shortest path of multiple queries with our proposed parallel-shortest path algorithm between the source and the target. Using Spark, we run in parallel a varying number of queries, each computing shortest path between a (source, destination) pair; see Sect. 8.3 for details. In these experiments, we use Rhode Island (RI) road network, USA, which consists of 53,658 vertices and 69,213 edges.

The results in Fig. 8.7 show that parallelization does not have a significant impact when executing a small number of queries. This is because the job is too small compared to the number of cores. It has an ineffective job distribution and takes a long time for I/O overhead among the cores which are distributed among up to 10

nodes (with 24 cores each). Three different queries are used in the figure: 10, 100, and 1000 queries. The horizontal axis shows results for varying number of cores: 23, 46, up to 230. Each Aziz node contains 24 cores. However, we keep one core for the operating system to perform its job. Thus, we utilize 23 cores for each node. The vertical axis gives the total runtime to compute the whole sets of queries.

A larger number of queries (10 K, 100 K, and 1 M) show a clear reduction and advantage in execution time while parallelizing the whole sets of queries as shown in Figs. 8.8 and 8.9. As usual the letter K denotes a thousand and M indicates a million.

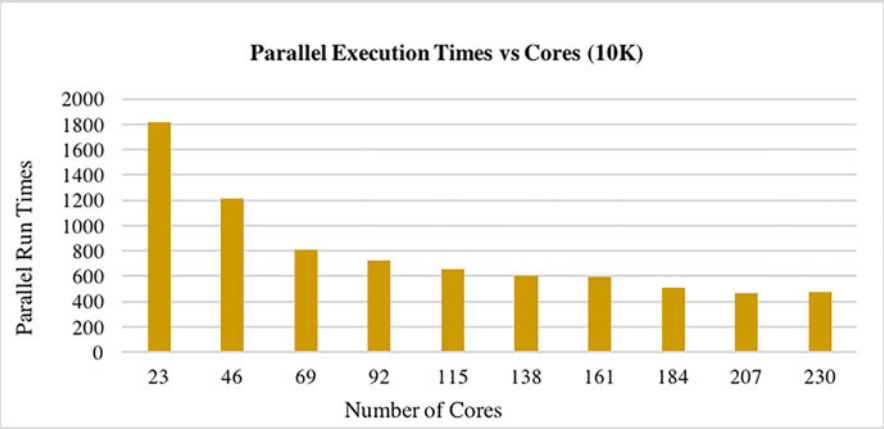


Fig. 8.8 Parallel execution time of varying number of nodes

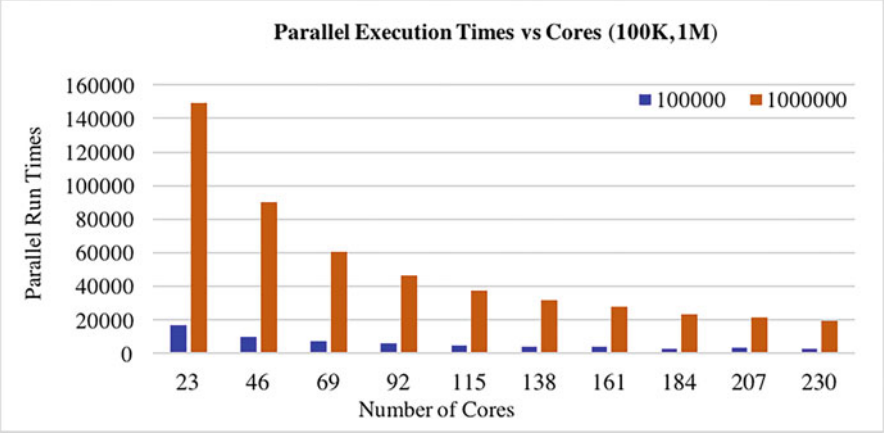


Fig. 8.9 Parallel execution time of varying number of nodes

8.4.3 Speedup

According to the experimental results in Sect. 8.4.2, we have calculated the achieved speedup. Figure 8.10 depicts that the achieved speedup is increasing with the increasing number of cores: 46 to 230. The figure depicts the speedups for six different query set sizes: 10, 100, 1 K, 10 K, 100 K, and 1 M. Note that the speedups for smaller computations get saturated for a smaller number of nodes compared to, for example, for larger query set of 1 M. The speedup is measured by using the following well-known formula.

$$Sp = \frac{T_s}{T_p}$$

Sp denotes the achieved speedup, while *Ts* denotes the execution time of the sequential computation, and *Tp* denotes the execution time of parallel computation.

8.4.4 Relative Speedup

To further elaborate the speedup saturation for increasing query set sizes and the number of cores, we now investigate relative speedup, the core-based speedup. The gained relative speedup is quite stable for large number of queries (1 M), and it is fluctuating for 100 K queries, as shown in Fig. 8.11. Whereas, for small queries less than 10 K, the relative speedup is decreasing. The following formula is used to calculate the relative speedup.

$$\text{Relative speedup} = \frac{Sp}{NC}$$

Sp and *NC* indicate the achieved speedup and the number of used cores, respectively.

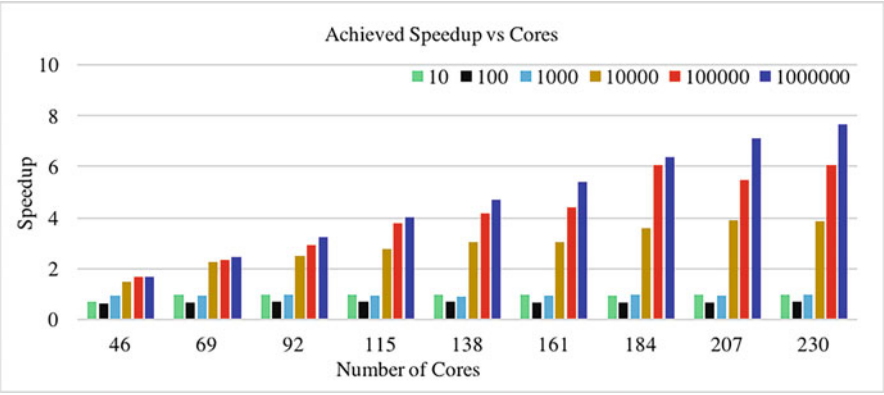


Fig. 8.10 Achieved speedup with different number of cores

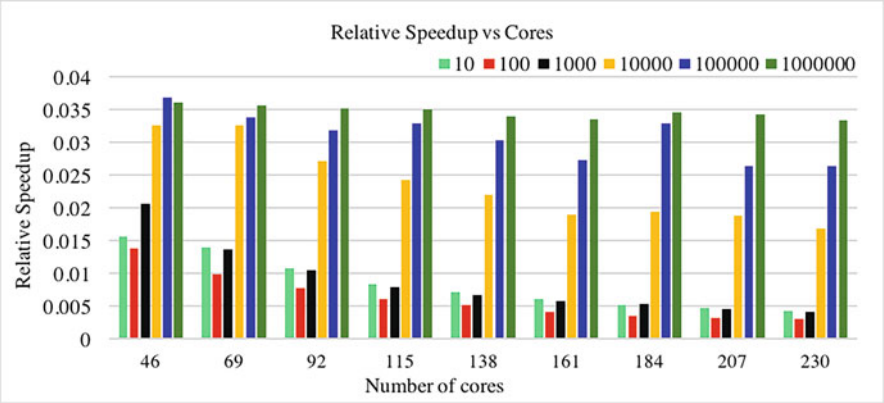


Fig. 8.11 Achieved relative speedup with different number of Aziz nodes

8.5 Conclusion

Smart applications and infrastructures are increasingly relying on graph computations to model real-life problems and process big data. The emergence of many graph-based software, programming languages, graph databases, and benchmarks, and their use in application domains provide the evidence for the increasing popularity of graph-based computing. In this chapter, we have our earlier work on single source shortest path computations of big data road network graphs using Apache Spark. In our earlier work [33], we had used the US road network data modelled as graphs and calculated shortest paths between two vertices over a varying number of up to 368 compute cores. The experiments were performed on the Aziz supercomputer (a former Top500 machine [34]). We had analyzed Spark's parallelization behavior by solving problems of varying graph sizes, i.e., various states of the USA with up to over 23 million vertices and 58 million edges. We call our system the Big Data Shortest Path Graph Computing (BDSPG) system.

In this chapter, we have focused on computing a set of large varying number of shortest path queries on a (source, destination) vertex pair. The number of queries used were 10, 100, 1 K, 10 K, 100 K, and 1 M, executed over up to 230 CPU cores. We achieved good performance, and as expected, the speedup is dependent on both the size of the data and the number of parallel nodes. In addition to the extended results, we have provided a detailed literature on shortest path graph computations. The system architecture for graph computing in Spark was explained with additional details using the architecture depiction and elaborated algorithms.

Future work will look into improving algorithms for sequential shortest path algorithm and its parallelization including data locality. There is a need for further performance analysis of our proposed system. We wish to apply the BDSPG system to the smart city case studies developed in [5, 6, 55].

Acknowledgments The authors acknowledge with thanks the technical and financial support from the Deanship of Scientific Research (DSR) at the King Abdulaziz University (KAU), Jeddah, Saudi Arabia, under the grant number G-651-611-38. The experiments reported in this chapter were performed on the Aziz supercomputer at King Abdulaziz University.

References

1. Lu, Y., Cheng, J., Yan, D., Wu, H.: Large-scale distributed graph computing systems. Proc. VLDB Endow. **8**, 281–292 (2014)
2. Sanfeliu, A., Alquézar, R., Andrade, J., Climent, J., Serratosa, F., Vergés, J.: Graph-based representations and techniques for image processing and image analysis. Pattern Recogn. **35**, 639–650 (2002)
3. Ding, Y., Yan, S., Zhang, Y., Dai, W., Dong, L.: Predicting the attributes of social network users using a graph-based machine learning method. Comput. Commun. **73**, 3–11 (2016)
4. Khan, A., Uddin, S., Srinivasan, U.: Adapting graph theory and social network measures on healthcare data. In: Proceedings of the Australasian Computer Science Week Multiconference on - ACSW '16. pp. 1–7. ACM Press, New York, New York, USA (2016)
5. Mehmood, R., Meriton, R., Graham, G., Hennelly, P., Kumar, M.: Exploring the influence of big data on city transport operations: a Markovian approach. Int. J. Oper. Prod. Manag. **37**, 75–104 (2017)
6. Mehmood, R., Graham, G.: Big Data Logistics: A health-care Transport Capacity Sharing Model. In: Procedia Computer Science. pp. 1107–1114 (2015)
7. Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.): Smart Societies, Infrastructure, Technologies and Applications, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST), Volume 224. Springer International Publishing, Cham (2018)
8. El-Gorashi, T.E.H., Pranggono, B., Mehmood, R., Elmirghani, J.M.H.: A data mirroring technique for SANs in a metro WDM sectioned ring. In: ONDM 2008 - 12th Conference on Optical Network Design and Modelling (2008)
9. Ayres, G., Mehmood, R., Mitchell, K., Race, N.J.P.: Localization to enhance security and services in Wi-Fi networks under privacy constraints. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 16. pp. 175–188. Springer (2009)
10. El-Gorashi, T.E.H., Pranggono, B., Mehmood, R., Elmirghani, J.M.H.: A mirroring strategy for SANs in a metro WDM sectioned ring architecture under different traffic scenarios. J. Opt. Commun. **29**, 89–97 (2008)
11. Mehmood, R., Pranggono, B., El-Gorashi, T., Elmirghani, J.: Performance evaluation of a metro WDM slotted ring network with san extension. In: Proceedings of the 7th IASTED International Conferences on Wireless and Optical Communications, WOC 2007. pp. 231–236 (2007)
12. Mehmood, R., Alturki, R., Faisal, M.: A Scalable Provisioning and Routing Scheme for Multimedia QoS over Ad Hoc Networks. (2009)
13. Mehmood, R., Alturki, R.: Video QoS analysis over wi-fi networks. Adv. Video Commun. over Wirel. Networks. 439–480 (2013)
14. Alturki, R., Mehmood, R.: Cross-Layer Multimedia QoS Provisioning over Ad Hoc Networks. Using Cross-Layer Tech. Commun. Syst. Tech. Appl. IGI Glob. Hershey, PA. 460–499 (2012)
15. Hendrickson, B., Kolda, T.G.: Graph partitioning models for parallel computing. Parallel Comput. **26**, 1519–1534 (2000)
16. Mehmood, R., Crowcroft, J.: Parallel iterative solution method for large sparse linear equation systems. Technical Report Number UCAM-CL-TR-650, Computer Laboratory, University of Cambridge, Cambridge, UK (2005)

17. Kwiatkowska, M., Parker, D., Zhang, Y., Mehmood, R.: Dual-processor parallelisation of symbolic probabilistic model checking. In: DeGroot, D., Harrison, P. (eds.) Proceedings - IEEE Computer Society's Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, MASCOTS, pp. 123–130. IEEE, Volendam, The Netherlands (2004)
18. Mehmood, R.: Disk-based Techniques for Efficient Solution of Large Markov Chains, PhD Thesis, School of Computer Science, University of Birmingham, (2004)
19. Mehmood, R., Parker, D., Kwiatkowska, M.: An efficient BDD-based implementation of Gauss-Seidel for CTMC analysis. Technical report CSR-03-13, School of Computer Science, University of Birmingham, Birmingham, UK (2013)
20. Eleliemy, A., Fayze, M., Mehmood, R., Katib, I., Aljohani, N.: Loadbalancing on Parallel Heterogeneous Architectures: Spin-image Algorithm on CPU and MIC. In: EUROSIM 2016, The 9th Eurosim Congress on Modelling and Simulation. p. 6. Oulu, Finland (2016)
21. Schlingensiepen, J., Mehmood, R., Nemtanu, F.C., Niculescu, M.: Increasing sustainability of road transport in European cities and metropolitan areas by facilitating autonomic road transport systems (ARTS). In: Wellnitz, J., Subic, A., Trufin, R. (eds.) Sustainable Automotive Technologies 2013 Proceedings of the 5th International Conference ICSAT 2013, pp. 201–210. Springer International Publishing, Ingolstadt, Germany (2014)
22. Junghanns, M., Petermann, A., Neumann, M., Rahm, E.: Management and analysis of big graph data: current systems and open challenges. In: handbook of big data technologies. Pp. 457–505. Springer international publishing, Champions (2017)
23. Altowaijri, S., Mehmood, R., Williams, J.: A quantitative model of grid systems performance in healthcare organisations. In: ISMS 2010 - UKSim/AMSS 1st International Conference on Intelligent Systems, Modelling and Simulation. pp. 431–436 (2010)
24. Tawalbeh, L.A., Bakhader, W., Mehmood, R., Song, H.: Cloudlet-based mobile cloud computing for healthcare applications. In: 2016 IEEE Global Communications Conference, GLOBECOM 2016 - Proceedings (2016)
25. Muhammed, T., Mehmood, R., Albeshri, A., Katib, I.: UbeHealth: a personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities. IEEE Access. **6**, 32258–32285 (2018)
26. Oh, S., Ha, J., Lee, K., Oh, S.: DegoViz: an interactive visualization tool for a differentially expressed genes Heatmap and gene ontology graph. Appl. Sci. **7**, 543 (2017)
27. Mehmood, R., Faisal, M.A., Altowaijri, S.: Future networked healthcare systems: a review and case study. In: Boucadair, M., Jacquenet, C. (eds.) Handbook of Research on Redesigning the Future of Internet Architectures, pp. 531–558. IGI Global, Hershey, PA (2015)
28. Arfat, Y., Aqib, M., Mehmood, R., Albeshri, A., Katib, I., Albogami, N., Alzahrani, A.: Enabling smarter societies through Mobile big data fogs and clouds. Procedia Comput. Sci. **109**, 1128–1133 (2017)
29. Xin, R.S., Gonzalez, J.E., Franklin, M.J.: GraphX: A Resilient Distributed Graph System on Spark
30. Gonzalez, J.E., Xin, R.S., Dave, A., Crankshaw, D., Franklin, M.J., Stoica, I.: GraphX: Graph Processing in a Distributed Dataflow Framework
31. Apache Spark GraphX, https://spark.apache.org/graphx/
32. Apache Spark, https://spark.apache.org/
33. Arfat, Y., Mehmood, R., Albeshri, A.: Parallel shortest path graph computations of United States road network data on apache spark. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 323–336. Springer, Cham (2018)
34. Aziz Supercomputer, Top500, https://www.top500.org/site/50585
35. Büscher, M., Coulton, P., Efstratiou, C., Gellersen, H., Hemment, D., Mehmood, R., Sangiorgi, D.: Intelligent mobility systems: Some socio-technical challenges and opportunities. In: Communications Infrastructure. Systems and Applications in Europe, Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST 16. pp. 140–152 (2009)

36. Ayres, G., Mehmood, R.: On discovering road traffic information using virtual reality simulations. In: 11th International Conference on Computer Modelling and Simulation, UKSim 2009. pp. 411–416 (2009)
37. Mehmood, R.: Towards understanding intercity traffic interdependencies. In: 14th World Congress on Intelligent Transport Systems, ITS 2007. pp. 1793–1799. ITS America, Beijing (2007)
38. Ayres, G., Mehmood, R.: LocPriS: A security and privacy preserving location based services development framework. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), LNAI, Volume 6279, Part 4. pp. 566–575. Springer (2010)
39. Elmirghani, J.M.H., Badic, B., Li, Y., Liu, R., Mehmood, R., Wang, C., Xing, W., Garcia Zuazola, I.J., Jones, S.: IRIS: An inteligent radio-fibre telematics system. In: Proceedings of the 13th ITS World Congress, London, 8–12 October (2006)
40. Suma, S., Mehmood, R., Albugami, N., Katib, I., Albeshri, A.: Enabling next generation logistics and planning for smarter societies. Procedia Comput. Sci. **109**, 1122–1127 (2017)
41. Suma, S., Mehmood, R., Albeshri, A.: Automatic event detection in smart cities using big data analytics. In: International Conference on Smart Cities, Infrastructure, Technologies and Applications (SCITA 2017): Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 111–122. Springer, Cham (2018)
42. Alomari, E., Mehmood, R.: Analysis of tweets in Arabic language for detection of road traffic conditions. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 98–110. Springer, Cham (2018)
43. Mehmood, R., Nekovee, M.: Vehicular Ad hoc and grid networks: Discussion, design and evaluation. In: 14th World Congress on Intelligent Transport Systems, ITS 2007. pp. 1555–1562. ITS America, Beijing (2007)
44. Gillani, S., Shahzad, F., Qayyum, A., Mehmood, R.: A survey on security in vehicular ad hoc networks. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 59–74 (2013)
45. Alvi, A., Greaves, D., Mehmood, R.: Intra-vehicular verification and control: A two-pronged approach. In: 7th IEEE International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2010. pp. 401–405 (2010)
46. Schlingensiepen, J., Nemtanu, F., Mehmood, R., McCluskey, L.: Autonomic Transport Management Systems—Enabler for Smart Cities, Personalized Medicine, Participation and Industry Grid/Industry 4.0. In: Intelligent Transportation Systems – Problems and Perspectives, Volume 32 of the series Studies in Systems, Decision and Control. pp. 3–35. Springer International Publishing (2016)
47. Schlingensiepen, J., Mehmood, R., Nemtanu, F.C.: Framework for an autonomic transport system in smart cities. Cybern. Inf. Technol. **15**, 50–62 (2015)
48. Alam, F., Mehmood, R., Katib, I.: D2TFRS: An object recognition method for autonomous vehicles based on RGB and spatial values of pixels. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 155–168. Springer, Cham (2018)
49. Alazawi, Z., Altowaijri, S., Mehmood, R., Abdljabar, M.B.: Intelligent disaster management system based on cloud-enabled vehicular networks. In: 2011 11th International Conference on ITS Telecommunications, ITST 2011. pp. 361–368. IEEE (2011)
50. Alazawi, Z., Abdljabar, M.B., Altowaijri, S., Vegni, A.M., Mehmood, R.: ICDMS: An intelligent cloud based disaster management system for vehicular networks. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), LNCS, Volume 7266. pp. 40–56. Springer, Vilnius, Lithuania (2012)
51. Alazawi, Z., Alani, O., Abdljabar, M.B., Altowaijri, S., Mehmood, R.: A smart disaster management system for future cities. In: Proceedings of the 2014 ACM international workshop on Wireless and mobile technologies for smart cities - WiMobCity '14. pp. 1–10. ACM Press, New York, New York, USA (2014)

52. Alazawi, Z., Alani, O., Abdljabar, M.B., Mehmood, R.: An intelligent disaster management system based evacuation strategies. In: 2014 9th International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2014. pp. 673–678 (2014)
53. Alazawi, Z., Alani, O., Abdljabar, M.B., Mehmood, R.: Transportation evacuation strategies based on VANET disaster management system. Procedia Econ. Financ. **18**, 352–360 (2014)
54. Aqib, M., Mehmood, R., Albeshri, A., Alzahrani, A.: Disaster management in smart cities by forecasting traffic plan using deep learning and GPUs. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, Volume 224. pp. 139–154 (2018)
55. Mehmood, R., Lu, J.A.: Computational Markovian analysis of large systems. J. Manuf. Technol. Manag. **22**, 804–817 (2011)
56. Arfat, Y., Aqib, M., Mehmood, R., Albeshri, A., Katib, I., Albogami, N., Alzahrani, A.: Enabling Smarter Societies through Mobile Big Data Fogs and Clouds. In: Procedia Computer Science (2017), 109, 1128
57. Quddus, M., Washington, S.: Shortest path and vehicle trajectory aided map-matching for low frequency GPS data. Transp. Res. Part C Emerg. Technol. **55**, 328–339 (2015)
58. Szucs, G.: Decision support for route search and optimum finding in transport networks under uncertainty. J. Appl. Res. Technol. **13**, 125–134 (2015)
59. Feng, L., Lv, Z., Guo, G., Song, H.: Pheromone based alternative route planning. Digit. Commun. Networks. **2**, 151–158 (2016)
60. Zeng, W., Church, R.L.: Finding shortest paths on real road networks: the case for a *. Int. J. Geogr. Inf. Sci. **8816**, (2017)
61. Malewicz, G., Austern, M.H., Bik, A.J.C., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: A System for Large-Scale Graph Processing. Proc. 2010 ACM SIGMOD Int. Conf. Manag. data. 135–145 (2010)
62. Yan, J., Tan, G., Mo, Z., Sun, N.: Graphine: programming graph-parallel computation of large natural graphs for multicore clusters. IEEE Trans. Parallel Distrib. Syst. **27**, 1647–1659 (2016)
63. Selim, H., Zhan, J.: Towards shortest path identification on large networks. J. Big Data. **3**, (2016)
64. Zhou, X., Chang, P., Chen, G.: An Efficient Graph Processing System. Asia-Pacific Web Conf. LNCS. 401–412 (2014)
65. Cao, Z., Guo, H., Zhang, J., Niyato, D., Fastenrath, U.: Finding the shortest path in stochastic vehicle routing: a cardinality minimization approach. IEEE Trans. Intell. Transp. Syst. **17**, 1688–1702 (2016)
66. Hou U, L., Zhao, H.J., Yiu, M.L., Li, Y., Gong, Z.: Towards online shortest path computation. IEEE Trans. Knowl. Data Eng. 26, 1012–1025 (2014)
67. Strehler, M., Merting, S., Schwan, C.: Energy-efficient shortest routes for electric and hybrid vehicles. Transp. Res. Part B Methodol. **103**, 111–135 (2017)
68. Hong, I., Murray, A.T., Rey, S.: Obstacle-avoiding shortest path derivation in a multicore computing environment. Comput. Environ. Urban. Syst. **55**, 1–10 (2016)
69. Mozes, S., Nussbaum, Y., Weimann, O.: Faster shortest paths in dense distance graphs, with applications. Theor. Comput. Sci. **1**, 1–25 (2014)
70. Abraham, I., Goldberg, A. V, Werneck, R.F.: A Hub-Based Labeling Algorithm for Shortest Paths in Road Networks. Springer-Verlag Berlin Heidelb. 2011. 230–241 (2011)
71. Sanders, P., Schultes, D.: Highway hierarchies hasten exact shortest path queries. Algorithms–Esa 2005. 568–579 (2005)
72. Peng, S., Sankaranarayanan, J., Samet, H.: SPDO: High-throughput road distance computations on Spark using Distance Oracles. 2016 IEEE 32nd Int. Conf. Data Eng. ICDE 2016. 1239–1250 (2016)
73. Zhu, A.D., Ma, H., Xiao, X., Luo, S., Tang, Y., Zhou, S.: Shortest Path and Distance Queries on Road Networks: Towards Bridging Theory and Practice. 857–868 (2013)
74. Zheng, C.Y., Wang, J.: All-Pairs Shortest Paths in Spark

75. Djidjev, H., Chapuis, G., Andonov, R., Thulasidasan, S., Lavenier, D.: All-pairs shortest path algorithms for planar graph for GPU-accelerated clusters. J. Parallel Distrib. Comput. **85**, 91–103 (2015)
76. Aridhi, S., Lacomme, P., Ren, L., Vincent, B.: A MapReduce-based approach for shortest path problem in large-scale networks. Eng. Appl. Artif. Intell. **41**, 151–165 (2015)
77. Faro, A., Giordano, D.: Algorithms to find shortest and alternative paths in free flow and congested traffic regimes. Transp. Res. Part C Emerg. Technol. **73**, 24–28 (2016)
78. Kajdanowicz, T., Kazienko, P., Indyk, W.: Parallel processing of large graphs. Futur. Gener. Comput. Syst. **32**, 324–337 (2014)
79. Liu, X., Zhou, Y., Guan, X., Sun, X.: A feasible graph partition framework for random walks implemented by parallel computing in big graph. Chinese Control Conf. CCC. 2015–Septe, 4986–4991 (2015)
80. Wang, Z., Chen, Q., Hou, B., Suo, B., Li, Z., Pan, W., Ives, Z.G.: Parallelizing maximal clique and k-plex enumeration over graph data. J. Parallel Distrib. Comput. **106**, 79–91 (2017)
81. Braun, P., Cuzzocrea, A., Leung, C.K., Pazdor, A.G.M., Tran, K.: Knowledge discovery from social graph data. Procedia Comput. Sci. **96**, 682–691 (2016)
82. Laboshin, L.U., Lukashin, A.A., Zaborovsky, V.S.: The big data approach to collecting and analyzing traffic data in large scale networks. Procedia Comput. Sci. **103**, 536–542 (2017)
83. Liu, R., Li, X., Du, L., Zhi, S., Wei, M.: Parallel implementation of density peaks clustering algorithm based on spark. Procedia Comput. Sci. **107**, 442–447 (2017)
84. Aridhi, S., Mephu Nguifo, E.: Big graph mining: frameworks and techniques. Big Data Res. **6**, 1–10 (2016)
85. Drosou, A., Kalamaras, I., Papadopoulos, S., Tzovaras, D.: An enhanced graph analytics platform (GAP) providing insight in big network data. J. Innov. Digit. Ecosyst. **3**, 83–97 (2016)
86. Zhao, Y., Yoshigoe, K., Xie, M., Zhou, S., Seker, R., Bian, J.: Evaluation and analysis of distributed graph-parallel processing frameworks. J. Cyber Secur. Mobil. **3**, 289–316 (2014)
87. Mohan, A., G, R.: A Review on Large Scale Graph Processing Using Big Data Based Parallel Programming Models. Int. J. Intell. Syst. Appl. 9, 49–57 (2017)
88. Miller, J.A., Ramaswamy, L., Kochut, K.J., Fard, A.: Research Directions for Big Data Graph Analytics. Proc. - 2015 IEEE Int. Congr. Big Data, BigData Congr. 2015. 785–794 (2015)
89. Chakaravarthy, V.T., Checconi, F., Petrini, F., Sabharwal, Y.: Scalable single source shortest path algorithms for massively parallel systems. Proc. Int. Parallel Distrib. Process. Symp. IPDPS. **28**, 889–901 (2014)
90. Xia, Y., Tanase, I.G., Nai, L., Tan, W., Liu, Y., Crawford, J., Lin, C.: Explore Efficient Data Organization for Large Scale Graph Analytics and Storage. Proc. 2014 IEEE BigData Conf. 942–951 (2014)
91. Zhang, M., Shen, F., Zhang, H., Xie, N., Yang, W.: Fast Graph Similarity Search via Locality Sensitive Hashing. Adv. Multimed. Inf. Process. PCM 2015. 9315, 447–455 (2015)
92. Pollard, S., Norris, B.: A Comparison of Parallel Graph Processing Benchmarks. (2017)
93. GraphX | Apache Spark
94. DIMACS Implementation Challenge, http://www.dis.uniroma1.it/challenge9/download.shtml
95. Gephi - The Open Graph Viz Platform, https://gephi.org/

Part II
Smart Healthcare

Chapter 9
A Survey of Methods and Tools for Large-Scale DNA Mixture Profiling

Emad Alamoudi, Rashid Mehmood, Aiiad Albeshri, and Takashi Gojobori

9.1 Introduction

According to The American Heritage Medical Dictionary, DNA profiling is "the identification and documentation of the structure of certain regions of a given DNA molecule, used to determine the source of a DNA sample, to determine a child's paternity, to diagnose genetic disorders, or to incriminate or exonerate suspects of a crime [1]." DNA profiling (also named DNA typing, DNA fingerprinting, or DNA testing) which was first introduced in 1985 by Alec Jeffreys has changed the area of forensic science significantly [2]. Dr. Jeffreys has found that there are several regions in the human DNA that contain repeated DNA sequence. He found that these DNA sequence areas may differ from one person to another. Dr. Jeffreys was able to measure the variation in these DNA sequences by developing a unique identity test called Restriction Fragment Length Polymorphism (RFLP). The repeated DNA areas are called Variable Number of Tandem Repeats (VNTRs).

E. Alamoudi (✉) · A. Albeshri
Department of Computer Science, Faculty of Computing and Information Technology (FCIT), King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: ealamoodi0004@stu.kau.edu.sa; aaalbeshri@kau.edu.sa

R. Mehmood
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: RMehmood@kau.edu.sa

T. Gojobori
Computational Bioscience Research Center (CBRC), King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia
e-mail: Takashi.Gojobori@kaust.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_9

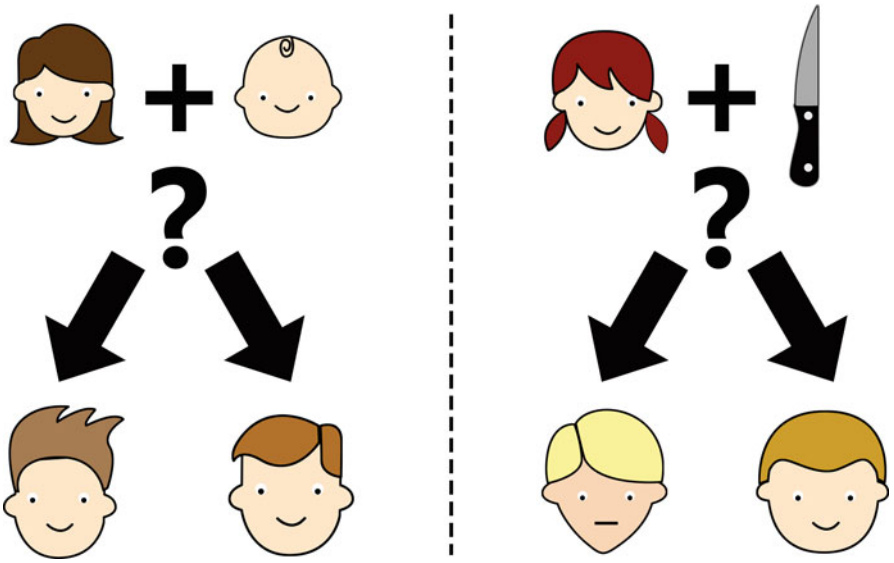


Fig. 9.1 DNA profile interpretation can have multiple usages, such as determine child's father and find a criminal among suspects

Today, DNA profiling is helping in many cases to identify an innocent from guilty. Human Identity test can also be used in contexts such as missing people investigation, parentage test, ancestry test, and disaster victim identification (see Fig. 9.1).

The DNA typing is considered today to be the most useful tool in the hand of law enforcement. Moreover, computer databases which contain DNA information of criminals which was taken from crime scenes had helped to associate a crime to an offender. Due to having a specific set of Short Tandem Repeat (STR) loci in these massive databases, it is unlikely to see a new set of DNA markers to be introduced shortly [2].

In order for a DNA sample to be processed, several steps should be considered [2]. First, obtaining the DNA from a biological source. Second, assessing the amount of DNA recovered. Third, isolate the DNA from its cells by using Polymerase Chain Reaction (PCR), which is a technique for copying specific DNA areas. Finally, the STR alleles which have been generated from the previous step will be examined. Figure 9.2 shows the steps used in DNA sample processing.

However, many difficulties may occur during the procedure of producing a DNA profile that affects the analysis of the sample. One of these problems is the stochastic effects, which arise during DNA extraction. Other challenges are allele drop-out, PCR process, allele sharing, and PCR amplification artifacts. Such difficulties hardened the accurate interpretation of the DNA profile [3].

The result of the DNA sample processing will be compared to other sample or databases to check the similarity. If there is a match or "inclusion," this indicates

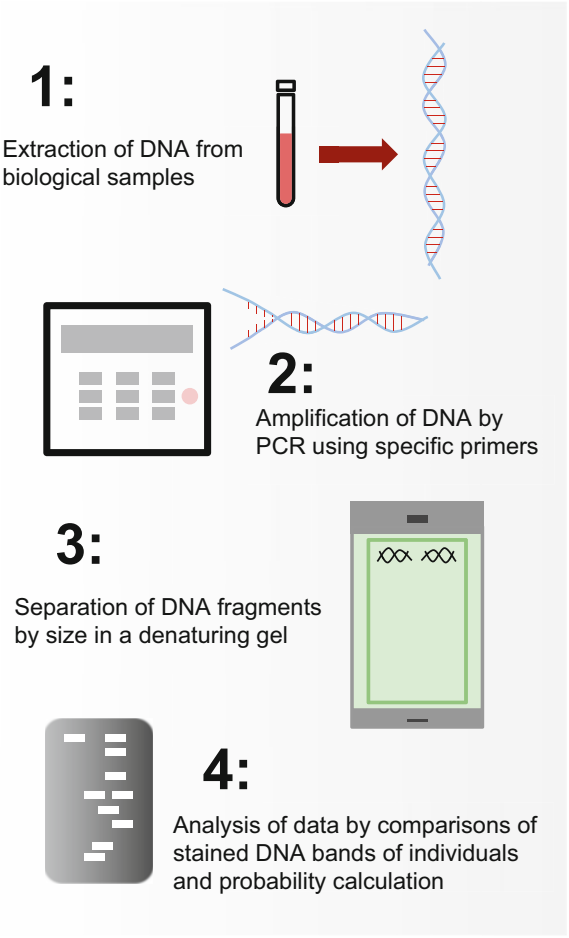


Fig. 9.2 The needed steps for DNA sample processing

that both samples were taken from the same source. On the other hand, if there is no match, the result would consider as "exclusion," which means there is no biological relation between the two samples [2]. A case report will be made by a forensic specialist explaining the result and containing random match probability answering the similarity question.

The Scientific Working Group on DNA Analysis Methods (SWGDAM) advise forensic report to contain a prediction of the number of contributors to the mixture that is under examination [3]. Usually, the number of contributors of a sample that taken from a crime scene is unknown. Therefore, an analyst should estimate it according to the electropherogram obtained. This assumption affects the final weight of DNA evidence [3].

In this chapter, we provide an extended review of DNA profiling methods and tools with a particular focus on their computational performance and accuracy. This is an extended version of our earlier work [4]. We have added further elaborations on the DNA profiling methods including DNA biology and genetics. Also, we discuss different HPC systems, namely, cloud, clusters, GPUs, and FPGAs. A background on parallel computing, MPI, OpenMP, and Java multithreading has been added. Additional DNA profiling tools have been reviewed and further explanation on the existing tools is provided. To the best of our knowledge, this is the first review work on DNA profiling tools.

Faster interpretations of DNA mixtures with a large number of unknowns and higher accuracies are expected to open up new frontiers for DNA profiling in the smart societies era. In the coming years, the complete genome sequencing technologies in a single or only a few cells will be easily available. These technologies may change the situation of DNA profiling completely. In this case, it is obvious to prepare appropriate statistical methods for that. It will be, therefore, important to prepare the mathematical and statistical algorithms for complete-genome-sequencing-based DNA profile. Emerging computational and big data developments [5], along with Internet of Things (IoT) [6] and smart society environments [7], will provide opportunities for new services related to DNA profiling.

The rest of the chapter is organized as follows. Section 9.2 describes background concepts related to this chapter including a background on DNA concepts, DNA profiling, parallel and High-Performance Computing (HPC). Section 9.3 discusses several methods for evaluating the DNA mixture statistically. Section 9.4 describes a number of approaches that rely upon the calculation of likelihood ratio to interpret DNA profile. We further discuss the importance of the Number of Contributors (NoC) in profiling a DNA mixture in Sect. 9.5. Some implementations that estimate the NoC was mentioned in the same section. Section 9.6 then illustrates notable DNA profiling tools. We conclude and give an outlook for the future of DNA profiling in Sect. 9.7.

9.2 Background Material

We now give a brief background of the various concepts and methods related to DNA profiling. The list of topics covered are DNA biology and genetics, forensic science, DNA mixture and its technologies, genetic markers, factors that increase the complexity of DNA profiling, likelihood estimator, the use of HPC in bioinformatics field, and HPC system and parallel frameworks.

9.2.1 DNA Biology and Genetics

The basic unit of living species is the cell, which produces energy and raw materials. To keep a cell operating, thousands of proteins are required. An individual body usually contains 100 trillion cells [2]. All these cells come from a single cell called zygote, which is formed from the merging of the mother's egg and the father's sperm. All cells share the same genetic sequences. Inside the nucleus of the cell is a chemical substance called DNA, which encodes protein construction data and cell replication information.

DNA, or Deoxyribonucleic Acid, is acting like a blueprint for our bodies since it contains all the required information for passing down genetic attributes to next generations. The entire DNA of a cell is called a genome.

DNA serves two essential purposes: first, makes replication of itself; second, handles information about protein producing instructions. Its alphabet contains only four letters: Adenine (A), Thymine (T), Cytosine (C), and Guanine (G) [2]. These letters are known as nucleotides or bases. Different combination of these bases can make the difference between humans and other species. The human body contains around three billion nucleotides. Each nucleotide is linked to its complementary base through hydrogen bonds that link the bases. The complementary base for adenine is thymine, and it cannot pair up with either cytosine or guanine. On the other hand, cytosine can only pair up with guanine. Moreover, there are three hydrogen bonds that connect cytosine and guanine, and two bonds linking thymine and adenine. Therefore, the C-G base pair is a bit stronger than the A-T ones [2].

DNA is composed of two twisted strands, or double helix, each of which comes from both parent. The DNA is divided into chromosomes; each chromosome acts like a container for the DNA molecule in a thread-like structure. A human genome is made up of 46 chromosomes or 23 pairs of chromosomes. Out of these 23 pairs, 22 pairs are autosomal chromosomes and one pair of the chromosome is for sex determination. Males will have X and Y chromosomes, whereas females will have two X chromosomes. Autosomal chromosomes are frequently used in human identity test [2], while the sex determination chromosome is usually used for sex determination tests.

A cell is called haploid if it contains only one set of chromosomes, like gamete cell (sperm and egg) However, if two sets of chromosomes do exist, a cell then is called diploid [2]. Triploid and tetraploid refer to having three or four sets of chromosomes, respectively.

A chromosome will have coding and noncoding areas: coding areas, or gene, are the regions that have the essential information for protein construction for cells. A gene size range between a few thousand and tens of thousands of base pairs [2]. A one-to-one comparison between biological and printed terms is presented in Fig. 9.3.

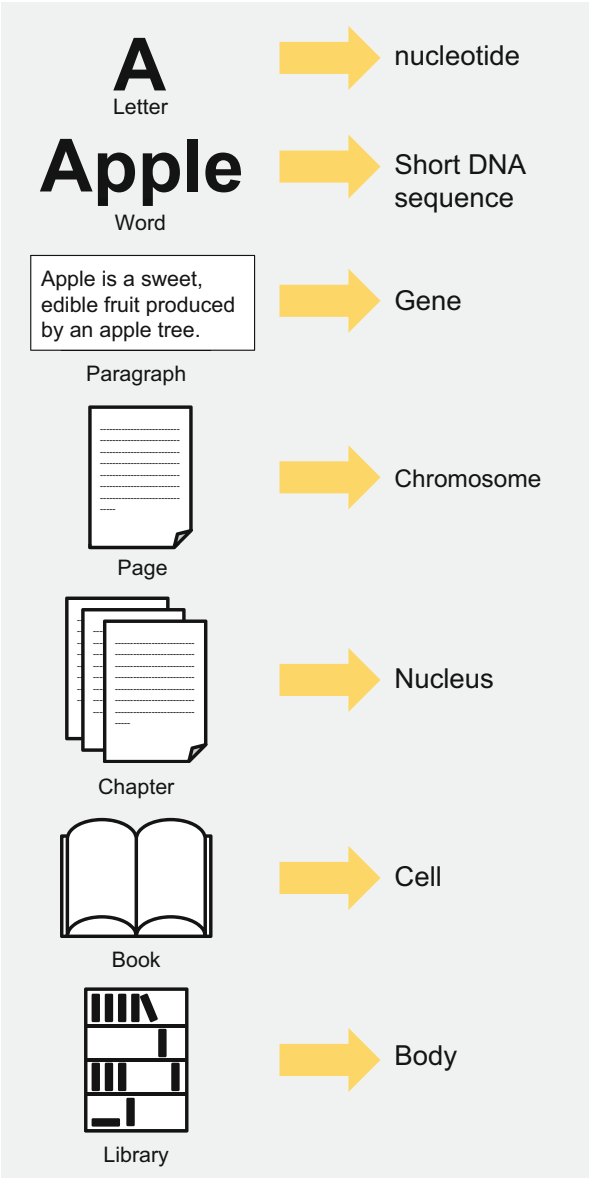


Fig. 9.3 Comparison between printed and genetic terms

9.2.2 Forensic Science

Forensic DNA tests had a major influence on the evolution of the criminal justice system. Yet, the advancement of new technologies is enabling forensic labs to expand its capabilities and improved the sensitivity of the DNA interpretation.

Butler [8] thinks that this area would develop in the future in three main areas; DNA technologies will become faster, the sensitivity of extracting relative information will increase, and higher volume of data will be expected due to that sensitive nature. He argued that STR will remain the dominant genetic marker.

According to Butler [8], key challenges in the forensic science field are the subjectivity, inconsistency of the complex DNA mixture interpretations between different laboratories and analysts, and the need for training forensic analyst to enhance interpretation of DNA profiles.

9.2.3 DNA Mixture

A sample is called a DNA mixture when two or more individuals contribute to it. Under some circumstances, the interpretation of a mixture could be more challenging. Allele sharing is one of the factors that increase the difficulty of interpreting a profile [2]. If we have a two-person mixture, then we expected to observe only four alleles per locus. However, this rule may change if we have alleles overlapping or if we have heterozygous individuals. If we have more than four alleles per locus, then we might deal more than two people mixture [9].

DNA mixtures interpretation is a very demanding task [10]. Perez et al. define the DNA mixtures as when two or more people contribute to the same sample. They added that contributors include victims, perpetrators, or other people who interact with the crime scene. Yet, the mixture can be complex when it became a subject of allele drop-in or/and allele drop-out [11]. A detailed introduction to the DNA analysis on the forensic science domain was given by [2, 12]. Butler gives a historical overview explaining the evolution of the area. He also explains the structure of the DNA and its fundamental component.

9.2.4 Technologies for DNA Profiling

The topic of DNA profiling was improved by the new advances in the technology. Weedn and Foran [12] gave a general overview of the latest updates and challenges in the forensic science domain related to DNA profiling. STR followed by PCR amplification is one of the most used methods that regularly used in forensic labs [12]. Other markers such as Single Nucleotide Polymorphisms (SNP), Y chromosome STRs, and mitochondrial DNA are also considered. Weedn and Foran

argued that the forensic DNA typing is the most dominant method in the forensic science laboratory. They mentioned that the forensic test usually performed with taking into consideration the court challenges. Therefore, the forensic science only uses a well-validated procedure, and all the laboratory processes should be documented. The protocols should be ready to be defended against legal attacks.

New technologies had not only increased the quality of profiling the DNA mixture, but also amplified artifacts such as stutter, variabilities, and baseline noise. Monich et al. [13] had introduced a quantitative signal model which forms the variability in a stutter, baseline noise, and allele peak height. They had also applied the chi-squared and Kolmogorov-Smirnov (KS) tests on the true peak heights and noise to test the fitness of various probability distribution classes. They argued that the interpretation of signal measured from a DNA sample used to be accomplished by using thresholding. Nonetheless, using thresholds during DNA analysis might lead to losing valuable information. For that reason, new methods that don't rely on threshold were developed.

9.2.5 Genetic Markers

Many genetic markers are used for mixture analysis such as restriction fragment length polymorphism (RFLP), STR, SNP, Y chromosomes, and mitochondrial DNA (mtDNA). The number of contributors in a mixture can be identified by counting the number of Y-STR alleles [14]. mtDNA can be used to determine the number of contributors and also it can be used with degraded specimens.

RFLP The restriction fragment length polymorphism (RFLP) was a popular DNA analysis during the 1980s [12]. RFLP was introduced by Dr. Edwin Southern in 1975. It involves too much work, yet it reveals only a little. Therefore, it was replaced by other techniques which were more robust, sensitive, and affordable.

STR STR marker has been used for DNA mixture analysis for many years. Available commercial tools offer limited STR markers, which give limited statistical support for the inclusion of mixtures. Therefore, Y chromosome STR analysis has been introduced to give extra means for the analysis of mixtures in forensic cases.

SNP SNP is a genetic variation among individuals. It appears throughout a person's DNA. In a diploid human genome, which consists of around six billion base pairs, there are almost 15 million SNP sites [12]. However, this method has many problems. For example, it cannot be used when the suspect is unknown. Moreover, SNP is not compatible with STR databases, and establishing SNP database would require extra work [14].

mtDNA Mitochondrial DNA analysis is used in cases when tissues are lacking a nucleus. Since it is present at a high copy number in each cell, it has been used with highly degraded specimens. In forensic labs, mtDNA is wildly used to analyze shed hair that lack roots [12]. In addition, it can be used with fingernails and keratotic

Table 9.1 A comparison between DNA typing methods in forensic labs

DNA interpretation method	PCR-based	Date of introducing	Usefulness
RFLP	✗	1980s	Regular caseworks
STR	✓	1980s	Regular caseworks
SNP	✓	2000s	Extremely degraded sample
Y-chromosome	✓	2000s	Vaginal swabs in rape cases
mtDNA	✓	1990s	Degraded sample and hairs

It was inspired by [12]

skin. However, forensic labs do not highly adopt mtDNA because it depends on DNA sequencing, which is labor-intensive, slow, and expensive process [12].

The Spanish and Portuguese Working Group of the International Society for Forensic Genetics (GEP-ISFG) made a considerable effort toward standardizing and improving the accuracy of the mtDNA analysis.

Table 9.1 shows a comparison between some DAN profiling methods. The first column describes the genetic marker. Column 2 specifies whether or not the genetic maker is PCR-based. Column 3 states when the genetic marker starts to be active. The last column shows how the genetic marker can be used.

9.2.6 Factors Increasing the Complexity of DNA Profiles

Different phenomena affect the complexity of interpreting a DNA profile. These factors include: the number of contributors, peak heights, stutter, a major peak masking, a stutter peak masking, population, drop-out probability, drop-in probability, and analytical threshold. No software had yet considered all these factors in its calculation [15]. Therefore, it is part of the challenges that face people who develop DNA mixture analysis tools to select which factor to model in their implementation.

9.2.7 Likelihood Estimator

Likelihood ratio (LR) is the probability comparison between evidence under two propositions [2]. One is called the prosecution hypothesis, which assumes that the DNA collected from a crime scene goes to the suspect, whereas the other is the defendant hypothesis, which assumes that the matches between the suspect and the questioned sample happened coincidentally. The two considered propositions are mutually exclusive.

The likelihood ratio is calculated by putting the prosecution hypothesis as a numerator while putting the defendant hypothesis as a denominator [2]. The LR equation is:

Table 9.2 The strength of evidence according to LR result [2]

Likelihood ratio	Corresponding evidence
1 to 10	Limited support
10 to 100	Moderate support
100 to 1000	Moderate strong support
1000 to 10,000	Strong support
10,000 or greater	Very strong support

$$LR = Hp/Hd \tag{9.1}$$

If we assume that the suspect commits the crime (100% probability), which is the prosecution hypothesis, then $Hp = 1$. Additionally, if the STR typing result is heterozygous, the probability of the defendant hypothesis would be $Hd = 2pq$, where p and q are the occurrences of the allele one and two for a locus in a relevant population [2]. If we have a homozygous STR typing, then the probability of the defendant hypothesis would be $Hd = p^2$. Therefore, the equation would become:

$$LR = Hp/Hd = 1/2pq \tag{9.2}$$

Butler [2] said that if the final result was greater than one, then this result would support the prosecution side. While if it is less than one, then the defendant theory would be in favor.

Typically, the LR will have a higher ratio if the STR genotype is rear because of the reciprocal relationship. LR is the inverse of the locus estimated frequency [2]. Note that the likelihood ratio can be more complex depending on the mixture of the evidence.

The strength of the result of the likelihood ratio in terms of the prosecution's case can be interpreted numerically as presented in Table 9.2. Column 1 represents the LR value, while Column 2 is showing the corresponding strength of evidence.

9.2.8 HPC Systems

In this section, we will explain four different types of HPC systems: FPGAs, clouds, GPUs, and clusters. Generally, FPGAs and GPU give better performance when algorithms are well designed, but they are extremely resource-constrained.

Cloud Usually, cloud NGS tools are built on the basis of the MapReduce framework [16]. Hadoop framework typically comes with MapReduce, and it distributes the work among compute cloud. MapReduce approach guarantees fault tolerance, load balancing, and redundancy. An example of a genome assembler that uses MapReduce framework is [17]. Nevertheless, privacy is still an issue when talking about cloud solutions.

Clusters Cluster HPC implementation usually combines Message Passing Interface (MPI) with another paradigm. MPI is used to distribute the task to other nodes (inter-node). On the other hand, the other paradigm usually takes care of the shared memory parallelism (intra-node). MPI + OpenMP is a common hybrid solution to perform fine- and coarse-grained optimization.

Optimize HPC implementation are much better than Hadoop solutions because fine-grained optimization is harder to achieve on Hadoop [16]. Consequently, Apache Spark was introduced to avoid Hadoop drawbacks. Still, well-tuned HPC implementation typically one order of magnitude faster than Apache Spark [16]. Apache Spark has the advantage of well-handling node failure and data replication.

A good future solution would combine HPC approaches and big data for processing NGS data. Such an approach has been successfully applied in domains such as machine learning [16].

GPU At its best performance, GPUs can give one order of magnitude better performance than CPUs [16]. CUDA is a programming language for general purpose applications runs at GPUs. Several NGS applications were successfully developed such as genome assembly [18], error correction [19], and k-mer counting [20].

However, developing an application to run on GPUs using CUDA requires a steep learning curve. It needs a deep understanding of GPUs architecture. As a result, very few tools have been targeting GPUs. Nevertheless, the new effort to develop highly optimized libraries such as NVBIO (https://developer.nvidia.com/nvbio) and the availability of languages like OpenACC might boost the GPUs effort in life science domain [16].

FPGAs FPGAs are chips that are able to be programmed that includes memory blocks and logic gates that can be configured manually. The configuration process usually is done through Verilog or VHDL programming languages [16]. FPGAs offer a highly scalable solution for NGS data. Example of FPGA-based tools includes FAssem assembler [21] and FADE tool for error correction [22]. Major drawbacks of using FPGAs-based are the long development cycle, and they are often not compatible to run on different FPGA generations. Yet, the new progress on higher level programming languages like OpenCL has smooth the way for the development of FPGAs-based solutions.

9.2.9 Parallel Frameworks

Parallel technologies are interesting on how to get the maximum benefit of the multicore/many-core processors and networked computing resources.

Many architectures have been proposed to enhance the resource utilization, namely, symmetric multiprocessor architecture (SMP), non-uniform memory access architecture (NUMA), simultaneous multithreading architecture (SMT), single instruction multiple data architecture (SIMD), and graphics processing unit (GPU).

In addition, multiple parallel programming frameworks have been suggested such as OpenMP, MPI, and MapReduce.

Various memory architectures exist, namely, shared memory, distributed memory, and hybrid memory architecture [23]. Shared memory systems enable all processes within the system to share memory as global memory space. In distributed memory systems, each processor has its own memory that cannot be reached by others, and no global address is available. They communicate, and send and receive data, through the network. Finally, hybrid memory systems combine both shared and distributed memory architectures. In clusters of multi-core or many-core processors, all processors within the machine shared their memory within each other; however, different machines can communicate over the network.

MPI Message Passing Interface (MPI) is a library specification for message passing model for distributed memory systems. It has multiple implementations such as OpenMPI, MPICH, and GridMPI [23]. Each processor, when using MPI, will have its own memory; moreover, it still can access other processors' memory using network communication. MPI offers point-to-point, from one processor to another, and collective communication, from one or many processors to one or many processors. MPI can send and receive message between processes in different modes, such as block and non-block communication. The message size can be in gigabytes [23]. MPI can run on many platforms like Windows, OS X, Linux, and Solaris. Programs written with the help of MPI can run on a single machine or a cluster of machines.

OpenMP OpenMP is an interface (API) for shared memory parallelism. It facilitates the programming process since it provides a set of directives for synchronization, parallelization, and managing the shared memory among threads.

When compiling a software written using OpenMP, multithreaded programs will be generated. Then, threads will share the memory address which will smooth the communication among threads.

OpenMP helps software developers to build parallel programs without in-depth knowledge of multithreading mechanism. Fine-Grained parallelism can be maintained over the OpenMP directives. Multiple languages support OpenMP such as C, C++, Fortran, Java, and it can run on multiple platforms like UNIX, LINUX, and Windows.

Java Multithreading Java supports multithreading shared memory parallel program language, which enables developing parallel software [24]. Multithreading feature in Java allows the execution of more than one part of a program concurrently to achieve better utilization of the computer resources. This can be achieved in Java through two ways: (1) extend the thread class, (2) by using the runnable interface [24]. One process can have multiple threads that share the same address space. Thus, a synchronization mechanism is vital to ensure data protection. Java implicitly maintains synchronization by using a lock for each object [24].

Java also provides a parallelization through distributed memory system by using API called MPJ, MPI equivalent for Java. MPJ allows developing a parallel software to run on a cluster system [25].

9.2.10 High-Performance Computing in Bioinformatics

Bioinformatics is a field that deals with massive data. Such data may require an extended time frame to be processed. Therefore, high-performance computing can help in shortening the time needed to finish the data processing. Perez et al. [10] discuss how HPC can help in solving bioinformatics problems. Authors had agreed that using advanced technologies had enabled remarkable discoveries in the medical field. They discussed different HPC systems which are used in bioinformatics area such as GPU computing. Graphics Processing Units (GPUs) are used to increase the computational capabilities of a group of PCs at a lower price. Moreover, they mentioned some HPC implementations in the bioinformatics field. These applications include Virtual Screening, Parallel Processing of Microarray Data, and Big Data Analytics and Network Models. In the end, authors had mentioned some drawbacks in the current HPC domain such as the energy consumption which can be overcome by using the virtualization concept, which enable sharing system hardware among different users. Other problems are the total cost of ownership and the high learning curve in upcoming programming models to influence their computational power.

Memeti et al. [26] had analyzed a DNA sequence on a heterogeneous platform that works with the Intel Xeon Phi coprocessor. These heterogeneous platforms usually come with one or more Xeon Phi devices and one or two general purpose CPUs. Researchers had introduced a parallel algorithm which can assign the workload of DNA sequence analysis to the different Xeon devices and host general purpose CPUs. This parallel implementation was aiming to reduce the overall analysis time. They also introduced a machine learning method that can predict the performance of the proposed algorithm on both the host and device. Finally, they evaluated the performance of their proposed method using human and animals' DNA on a platform that consists of an Intel Xeon Phi 7120p device with 61 core and two 12-core Intel Xeon E5 CPUs.

Bell and Gray [27] had given an overview of the history of supercomputer since the 1960s. Moreover, they tried to predict the future and how the next trend would be. They illustrated 50 years of evaluation in the high-performance computing domain. Authors argued that in 2001, there existed two major types of architectures: clusters of scalar multiprocessors and clusters of Cray-style vector supercomputers. They said that in the 1960s, Seymour Cray had proposed a parallel instruction implementation using parallel and pipelined function units. In 1982, Cray's research had reached to the multiprocessor (XMP) structure which helped to introduce the current supercomputer architecture. This architecture was sharing 10% of the market in 2001. However, a single node had reached its limit. So, to

go beyond that, a cluster architecture was proposed. In the 1980s, a cluster by CMOS-based killer micros had overcome the single node by better performance, scalability, and lower price. In 1993, NASA was looking for a supercomputer that satisfies its need which was 1 Gflops workstation. To achieve that, a Beowulf project was established which cost $40,000. In 2001, 28 Beowulfs were among the Top500 fastest supercomputers. In the end, authors had expected that there would be two possible paths for supercomputers to evolve in the future. One is an application-centric vector supercomputer. While the other concentrate on peta-scale datasets where users can get access to data.

Diegoli et al. [28] had estimated the recombination rate among 15 X STR markers by using data of genotype from 158 families and following earlier suggested a likelihood-based method which allows for single-step mutation. The computational challenges from the previous study were overcome by introducing a multi-core parallelization on the HPC system. Authors had argued that X STR is useful in forensic science due to a number of features such as their ease of haplotype inference because of the male hemizygosity and their particular mode of inheritance. They also added that few studies had systematically estimated the recombination rate among X STRs. Nonetheless, none of these studies had been comprehensive as their study.

To write an algorithm that can utilize an HPC system, a person should be able to deal with parallel programming languages. However, when writing an algorithm, different bugs may occur. Laguna et al. [29] had described the latest updates in designing a saleable debugging tool. They argue that debugging a parallel program is more difficult than debugging a serial one. Authors had focused on three dynamic debugging methods in both parallel programs and MPI instructions. The first dynamic approach is discovering scaling bugs, which helps to find bugs that are latent at a small scale while manifesting themselves at a larger scale. Vrisha is an example of this technique. Second, behavior-based debugging, this technique is based on observing the behavior of the processor. This helps to reduce the huge number of parallel processors into a small number of behavioral groups. AutomaDeD framework is a simple model of task behavior that saves information related to patterns and timing in each task's control flow. The information allows the developer to detect performance problems. Finally, software defects in MPI, MPI library implementations have suffered from software bugs, especially when ported to new machines. Many of these bugs are hard to find by average programmers. FlowChecker is an example of software that can detect MPI bugs. In the end, authors had focused their attention on three main problems that are still open in the domain which are programmability challenges, performance bugs, and detecting silent data corruptions.

In DNA profiling, the use of HPC has been limited. MPI has not been used. Most of the parallel tools have been developed using Java threads (e.g., LRmix Studio, CeesIt, and NOCIt), OpenMP (e.g., LikeLTD), and Snow parallel package in R (e.g., Kongoh and Euroformix). A distributed memory implementations of DNA profiling methods have not been reported to date.

9.3 DNA Profiling: General Methods

Several methods had been proposed to evaluate a DNA mixture statistically. Likelihood ratio, the combined probability of inclusion/exclusion (CPI/CPE), and a modified random match probability (mRMP) are some examples of these methods [30]. In February 2000, the FBI's DNA Advisory Board had strongly recommended the first two methods to be used [2]. Moreover, in 2006, the International Society of Forensic Genetics (ISFG) had emphasis on the value of likelihood ratio [30]. There are six steps to interpreting a DNA mixture which was first described by Tim Clayton in 1998 [2]. First, we need to identify the existence of a mixture. Second, the allele peaks should be selected. Third, we need to determine the possible number of contributors. Fourth, compute an approximation of the ratio of the people who contribute to the sample. Fifth, we need to calculate all potential genotype combinations. Finally, a reference sample comparison should be made.

In the CPI approach, an equal weight is given to all possible genotype combinations. Therefore, a lot of information is being wasted when using this approach which makes it inefficient when working with distinct genotypes [30]. This approach does not require prior knowledge of the number of contributors because it is evaluating all genotypes' combination based on the evidence profile [30].

The Random Match Probability (RMP), on the other hand, is usually used with single-source samples; therefore, a modified random match probability (mRMP) was proposed to deal with more single-source samples [30]. Unlike CPI, this approach requires prior knowledge of the number of contributors in the mixture and will not work well with low-level profiles. An example of two- and three-person mixtures calculations using mRMP was described in [31].

According to Bille et al., LR is the most dominant method of evaluating a DNA mixture. However, both mRMP and LR make use of the available information in the sample where CPI does not tend to do so.

More detailed analysis of the three methods and their advantages and weaknesses can be seen in Butler's book "Advanced Topics in Forensic DNA Typing: Interpretation" [30].

9.4 DNA Profiling Using Likelihood Ratio

LR is considered as the most appropriate and powerful approach for calculating the weight of DNA evidence. There are three methods using LR that are widely described in the literature. The first method is the binary model, which is the simplest yet it cannot handle complex mixture [32]. Second, the semi-continuous, which is the most used by scientists since it is easy to understand and explain, but it still neglects relevant information [33]. Finally, the continuous which overcomes most of the previous models' shortcomings. It utilizes most of the available information provided by the sample, yet it is harder to be accepted and explained in a courtroom

[32]. These models may involve a human or computerized process depending on the complexity of the approach. Kelly et al. [33] had made a comparison between these three approaches which are suggested by the DNA Commission of the ISFG.

Many frameworks that interpret complex DNA profiles rely on the likelihood ratios approach such as [11]. Gill et al. had mentioned a set of guidelines which can help to evaluate any complex mixture. In addition, they provide some features for any model that might deal with complex interpretation such as the ability to incorporate several contributors. They emphasize the fact that the calculation must be provided in a fast manner.

Most of the likelihood ratio-based analysis require the number of contributors to be given before the analysis start. For instance, [34–39] rely on the number of contributors on their analysis.

However, others had tried to avoid using it in their interpretation, such as [40, 41]. Russell et al. had developed a semi-continuous method that can calculate the likelihood ratios without previous knowledge about the contributor's number. Their simple model has the abilities to calculate the statistical weight to inclusions. They had also provided a limit test which will guarantee the absence of any false inclusion by chance. To test the proposed unconstructed likelihood ratio (UCLR) model, researchers had collected a set of DNA mixtures with known contributors in different ratios. The result shows good performance on three people mixture. However, the performance becomes worse as the number of contributors increased.

9.5 Estimating Number of Contributors for DNA Profiling

Today, most applications that interpreted the DNA profile do require the number of contributors to be available as input [40]. Different methods have been developed to conclude the number of contributors in a DNA mixture. One of these methods is called Maximum Allele Count (MAC). This approach calculates the minimum number of contributors who might contribute to a sample by counting the observed alleles at each locus. Nevertheless, this method may not be valid to work in a complex mixture because of the complexity of allele sharing [42]. New methods that were proposed do not only rely on the number of observed alleles, but also on the frequencies of observing the allele in the population. Biedermann et al. [43] had developed a probabilistic method that performs a Bayesian network to conclude the number of contributors in DNA mixture. The new approach performs better than MAC with a degraded DNA sample and a higher number of contributors. Maximum Likelihood Estimator (MLE) is another method used to estimate the number of contributors. It tries to maximize the likelihood value of the DNA profile [44].

Haned et al. [45] had compared MAC and MLE. The efficiency of both methods had been analyzed and compared for identifying two to five-person mixtures. Three different situations were used to test both methods. First, when all contributors belong to the same population and when allele occurrences are known. Second, when allele occurrences are not known, which may occur in population subdivision.

Finally, a condition of partial profiles and how it could affect the estimation accuracy. MAC method is used to set the lower bound that can clarify the number of alleles in a mixture. Haned et al. believe that MAC is unreliable since there is a chance for allele sharing between people which called the masking effect. The result of the comparison supports the use of MLE when a mixture contains more than three contributors. However, when three or two people contribute to a mixture, MAC would perform better.

However, as the number of contributors increased the risk would increase. Haned et al. [46] had analyzed the risk of dealing with three-, four-, and five-person mixture. They have done that by comparing the gold standard LR to the casework LR. The gold standard LR is when the number of contributors and genotypes are known which means the availability of all required information to compute LR per contributor. Authors showed the result and the implied thoughts of analyzing high order mixture in the forensic domain. Haned et al. argued that the low template DNA mixture of three-, four-, and five-person are common in forensic casework, yet it is hard to interpret.

Many methods are used today to evaluate the number of contributors in a sample such as [3, 9, 10, 47]. Perez et al. had created a strategy that could find out the number of contributors from two to four-person mixtures for both low template and high template DNA amounts. The proposed strategy helped to provide a useful tool to differentiate between high and low template two-, three-, and four-person mixtures. The four-person mixtures show some difficulties due to the allele sharing phenomena.

Egeland et al. focus on calculating the number of contributors in a mixture by maximizing the likelihood. The proposed approach is based on single SNP. The method tried to answer two questions: Is it a mixture? And if yes, then how many markers are required and how they should be selected. One of the recommendations that was driven from the result was regarding the number of markers needed to calculate the number of contributors which is 100 markers.

A typical algorithm for finding the best allele pair in a locus to interpret a mixture is presented in Algorithm 9.1. Such a process is essential when calculating the number of contributors in a DNA profile. Moreover, it is considered as a performance bottleneck.

On the other hand, Marciano and Adelman [48] proposed a machine learning approach that can estimate the number of contributors in a mixture. Their approach can handle mixtures with up to four contributors. The testing phase of this method shows a good result. The model first will be trained on a set of data, then it will be able to guess the number of contributors in DNA sample correctly. According to Marciano and Adelman, such a problem perfectly fits the domain of machine learning. The abundance of human mixture data can help to train the model well.

Yet, the machine learning approach suffers from several drawbacks. First, the quality of the result depends on the trained data. Uncorrected data may harm the system and lead to faulty results. Second, a training phase is always required before using the system. Such a phase is time-consuming and it might need to be redone many times. Third, the accuracy of the system starts to shape up after working

Algorithm 9.1. calculate locus's best allele pair that give best interpretation of the sample

```
1: procedure GeneProbCalc(stepSize, noc, lname, revLoci, forLoci, DNAmass, AlleleAtLoci, LDO)
   //noc=number of contributors, lname=locus name, LDO= Locus Drop Out
2:     locAlleles = AlleleAtLoci[locusname]
3:     MeanAndStd = Meanstd(locusname) //find mean and stddev
4:     for i=0 to stepSize do
5:         g=random array between 0 and 1 with size noc
6:         for j=0 to noc do
7:             for k=0 to 2 do
8:                 r=Generate random number that does not exceed the interval of the locus
9:                 allele = AlleleRange[r] //get the allele in the selected interval for a specific locus
10:                Add allele to Peakscumulative
11:                contMass=g[i-1]*DNAmass
12:                if Rand() < ExpVal(locusName, LDO, contMass) then
13:                    ValidAlleles.add(allele)
14:                    weight[allele] = weight[allele] + contMass
15:                end if
16:            end for
17:        end for
18:        for aName=ValidAlleles.start to ValidAlleles.end do //aName=Allele Name
19:            if locAlleles contains allele then
20:                (mean, variance) =Meanstd(weight[allele]) //find the mean and stddev
21:                if revLoci[lName] && Rand()<ExpVal(lname, RevStutDropO, weight) then
22:                    rMu=ExpVal2(lName, mean, weight[allele]) * allele.height
23:                    rSigma=ExpVal2(lName, Stddev, weight[allele]) * allele.height
24:                    revAlleleStut = aName - 10 // get the reverse
25:                    fowStutPeak = Peakscumulative[allele]
26:                end if
27:                means[revAlleleStut] = means[revAlleleStut] + rMu
28:                variances[revAlleleStut] = variances[revAlleleStut] + rSigma * r
29:                if forLoci[lName] && Rand()>ExpVal(lName, forStutDropO, weight) then
30:                    fMu=ExpVal2(lName, Mean, allele.weight) * allele.height
31:                    fSigma=ExpVal2(lName, Stddev, allele.weight) * allele.height
32:                    fowAlleleStut = aName + 10 // get forward
33:                    fowStutPeak = Peakscumulative[allele]
34:                end if
35:                means[fowAlleleStut] = means[fowAlleleStut] + rMu
36:                variances[fowAlleleStut] = variances[fowAlleleStut] + rSigma * rSigma
37:            end if
38:        end for
39:        for temp=Peakscumulative.start to Peakscumulative.end do
40:            mean.add(temp.allele, MeanAndStd[0])
41:            variances.add(temp.allele, MeanAndStd[1] * MeanAndStd[1])
42:        end for
43:        locusProb=calcLocusPeakHeightsProb(Peakscumulative, means, variances)
44:        Summation+ = locusProb
45:        if locusProb > currMax then
46:            currMax = locusProb
47:            for alleleName=selectedValidAlleles.start to selectedValidAlleles.end do
48:                currMaxAlls.add(alleleName)
49:            end for
50:        end if
51:    end for
52:    result.add(Summation, currMax, currMaxAlls)
53:    Return result
54: end procedure
```

Algorithm 9.1 A typical algorithm for calculating locus's best allele pair that gives the best interpretation which helps in finding the number of unknowns in a DNA mixture (algorithm inspired by NOCIT tool [3])

large data of human DNA mixtures. Such data may not be easily available. Finally, as the maximum number of contributors increase, the accuracy of the prediction will be declined. Authors said that they didn't go up to five contributors because misclassification of five contributors may occur on four contributors mixture [48].

9.6 Software Tools for DNA Profiling

A number of tools are available that implement various DNA profiling methods. These include DNA MIX [49], Euroformix [34], LRmix [36], LRmix Studio [32, 50], TrueAllele [35], LikeLTD [38], Lab Retriever [15], CeesIt [37], NOCIt [3], DNAMixture [51], Forensim [52], MixtureCalc, Mixture Analysis [53], FamLink kinship [54], DNA Mixture Separator [55], and STRmix [56]. We will review the most notable tools in this section. At the end of this section, we will provide a comparison between the selected tools.

9.6.1 DNA Mix

There are three versions of this software, and all of them are open sources. The third version is the most notable and powerful one among the three, and is based on [49]. This version is written in Java and is appropriate for complex mixtures as well as single-contributor stains. The software will ask for the database, stains, genotype, and hypothesis to be inputted.

On the latest version, dependency of all alleles was carried by contributors to the DNA mixture. All contributors will be assumed to belong to the same population, which will increase the effect that is being considered. Authors of DNA MIX did ignore the probability of null alleles. Thus, only homozygous contributors contribute a single allele to a profile. A simple GUI has been developed in this version (Fig. 9.4).

9.6.2 LRmix Studio

LRmix Studio is a software designed to interpret complex DNA profiles. It was built on its previous version, which called LRmix; however, LRmix Studio is much faster and more flexible. It can measure the probative value of any (autosomal STR-based) DNA profile [50]. It can handle uncertainty in the DNA mixture from the allelic drop-out and drop-in. Moreover, it is written in Java, and it is open source under the GPLv3 license (Fig. 9.5).

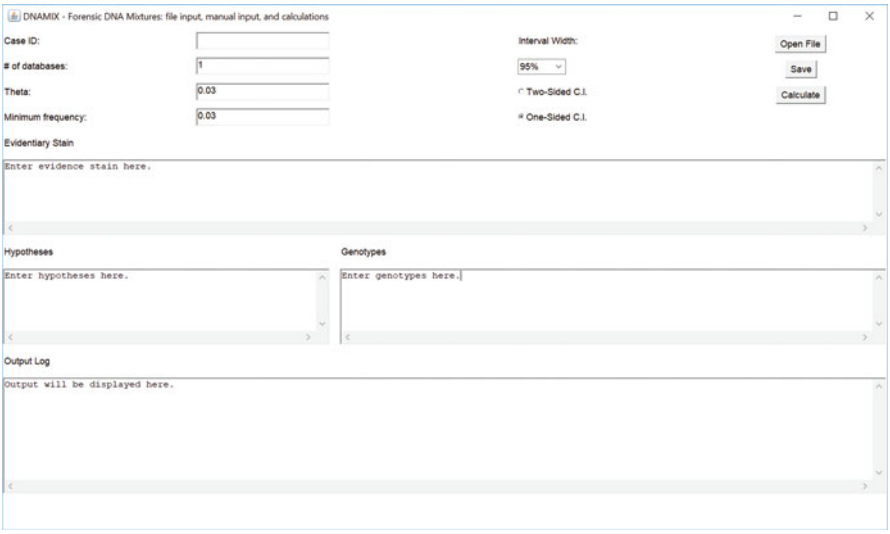


Fig. 9.4 The user interface for DNAMIX v3.2

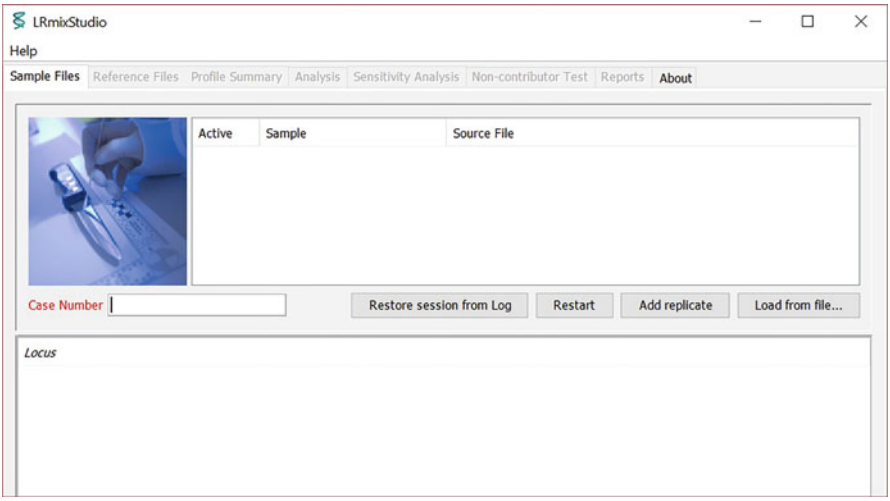


Fig. 9.5 The user interface for LRmix Studio v2.1.3

This software is following the semi-continuous model of interpreting DNA profiles. Both the prosecution and the defense hypotheses assume that contributors are unrelated. Yet, under the defense hypothesis, contributors can be related to an unknown contributor.

If there are missing data in the reference profiles, LRmix Studio tool will be unable to work properly. Moreover, it cannot deconvolute DNA profile because it does not explicitly include the information of the peak height.

9.6.3 TrueAllele

TrueAllele is a software that computes DNA interpretation automatically. It can infer genetic profiles from all sorts of DNA samples. The software applies the continuous model; however, no open source version of the code is available. It was written in Matlab. Analysis followed by a comparison of TrueAllele is presented on [35] using real information that has been taken from actual cases.

TrueAllele can separate complex DNA profiles into its component genotypes. For each locus for a given contributor, the genotype and the uncertainty of that genotype are labeled using the probability distribution over the potentials of the allele pair.

TrueAllele applies the MCMC (Markov Chain Monte Carlo) statistical search to sample from the joint posterior probability distribution. For each locus in every contributor, the posterior probability for the genotype is going to be calculated. Thus, to remove the examination bias, the genotype will be inferred exclusively from the evidence data [57].

9.6.4 Lab Retriever

Lab Retriever [15] is a free software developed to estimate the likelihood ratios that combine a probability of drop-out. It was built on the top of another software called LikeLTD which was written in R language. The front end of the software was developed using CSS, JavaScript, Python, and HTML. On the back end, authors rewrote the code using C++ to acquire more speed. The software uses the semi-continuous model. It computes likelihood ratios for up to four unknown contributors to a DNA sample.

Lab Retriever uses dynamic programming to speed up the computation, which will avoid iterating over all genotypes. This tool estimates the likelihood ratio and compares the evidence under various hypotheses, while still allow for drop-out of alleles.

In order for the system to work, the user must specify as an input the following: The detected alleles in the evidence profile, the suspect genotype, the genotype of other contributors, the considered hypotheses, and the database of allele frequency.

Moreover, several parameters should be specified such as the probability of drop-in and drop-out and the co-ancestry adjustment value (Figs. 9.6 and 9.7).

9.6.5 CeesIt

CeesIt (CEES: computational evaluation of evidentiary signal) [37] is a method that integrates two features of the continuous approach to calculate the LR and its distribution which are conditioned on the defense hypothesis and the linked

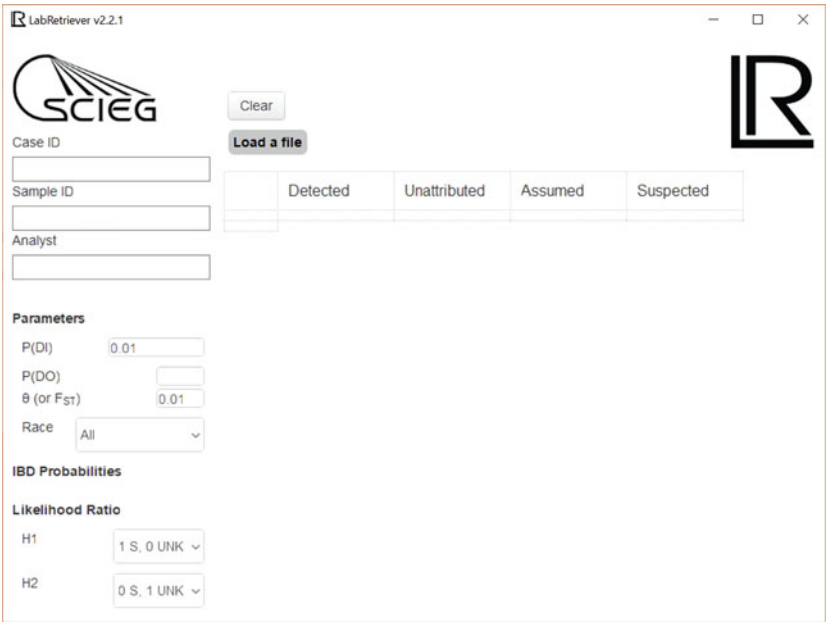


Fig. 9.6 The user interface for Lab Retriever v2.2.1

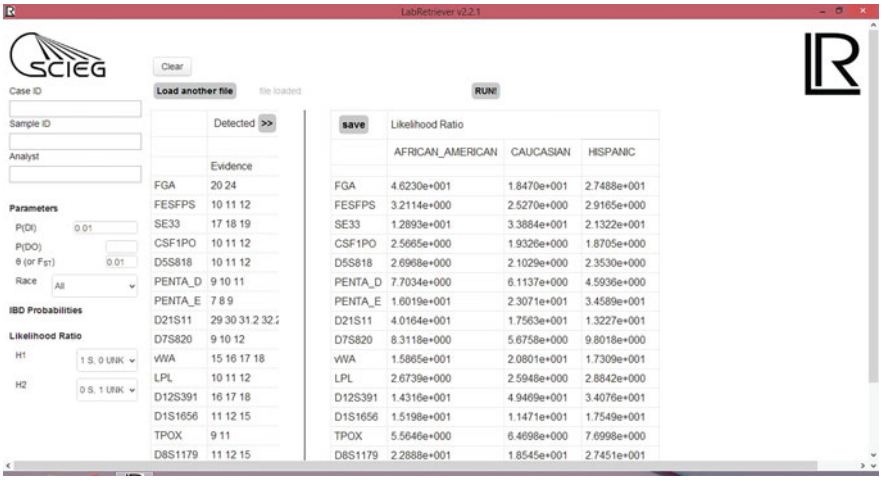


Fig. 9.7 Lab Retrievers v2.2.1 interface in action

Table 9.3 The running time of CeesIt under different number of contributors

Number of contributors	Average time (minutes)
1	7
2	50
3	140

p-value. It combines stutter, drop-out, and noise in its calculation. For calibration information, it uses a single-source sample with known genotypes. It calculates the LR for a selected Person of Interest (POI) on a questioned sample, together with the p-value and LR distribution.

To assess the performance of CeesIt, it was tested using 303 sample files ranging between one and three contributors, and the mass of the sample was ranging between 0.016 and 1 ng. The analysis results show a dependency on the number of contributors. Therefore, a good estimation for the number is critical for an accurate result.

The running time of the tool depends on the number of contributors. As the number increased, the time complexity will increase too. See Table 9.3 for more details on CeesIt running time.

Multithreaded is already implemented on CeesIt to increase resource utilization to acquire more speedup.

The software was written in Java and is available as a (.jar) file. An in-depth analysis of the software was presented on [37].

9.6.6 LikeLTD

LikeLTD is a software that is used for computing the likelihood of DNA profile evidence, including complex mixtures. It has been written in R. However, since the fifth version, the computation-intensive areas in code have been rewritten in C to be executed in parallel. This software applies the continuous model of calculating the Likelihood ratio. These areas include the computation of genotype combinations for unknown contributors, computing allele doses for each genotype combination, dose adjustments for relatedness, heterozygosity, drop-out, and power.

The runtime of the peak height model is much slower than the runtime of the discrete model, yet it yields a higher evidence weight (see Table 9.4). The time complexity of the peak height model scales up with the number of unknown contributors, the number of observed peaks, and the number of replicates in the

Table 9.4 The runtime of calculating the Weight of Evidence (WoE) using the two different models for the laboratory case [38]

Hypothesis	Model	WOE	Runtime (Minutes)
Q/X + K1 + U1	Discrete	2.3	14
	Peak height	8.2	23
Q/X + U1 + U2	Discrete	0.5	38
	Peak height	7.8	200

profile. Other parameters that increase the runtime are the modeling double-stutter or over-stutter. Parallelism was achieved on the C++ code by using a shared memory parallelism (OpenMP).

The runtime of the algorithms was recorded using a node with eight Intel Core I7 processors (3.1 Hgz per core) and with 15 Gb of RAM. The result is presented in Table 9.4. The first column describes the hypothesis that was applied. Two hypotheses were used. Q is a contributor to the crime scene profile under the Hp while X is the unknown individual under Hd that assumes to contribute to the profile instead of Q. The hypotheses may specify the number of K which represent the known contributors whereas U is the unknown contributors. The second column indicates the used model whether it uses discrete or peak height. The last two columns are showing the weight of evidence and the corresponding running time.

9.6.7 DNAMixture

DNAMixture is a statistical model that calculates and analyzes DNA sample for one or more contributors [51]. It uses Bayesian network representation to speed up the computation and allow analysis of mixtures which contain several unknown contributors. Alleles observing process is objective, and it does not depend on a subjective preprocessing of the DNA profile [58]. Such a preprocessing can lead to more errors. The model has been tested on some real case and the results were sensible and robust [58].

This software has been written in R and follows the "fully continuous" statistical model. Its authors claim to develop all methodology within a framework for consistent analysis and transparency. The application does not have a graphical user interface, which requires a basic experience in R. DNAMixture relies on an R package called "Hugin." Hugin is used to compute the Bayesian network. DNAMixture is not parallelized, yet the Hugin package is.

The computational complexity of the model depends on several factors. The running time of DNAMixture when there are five unknown contributors took 3 h on a regular desktop machine [58]. Authors claim that they perform analysis on several cases which takes 35 min; when they analyze the same cases using another tool called TrueAllele [57], the runtime goes to 36 h [58].

9.6.8 *Kongoh*

Kongoh [59] is an open-source application based on the continuous model for interpreting DNA sample. This model deals with artifacts and allelic drop-out ratio on its calculation, but it doesn't consider allele drop-in probability. It performs a Monte Carlo simulation based on the probability distributions of the given parameters. Next, gamma distributions will be used to approximate the peak heights that were generated by the simulation.

The number of contributors is not required to be given as an input. Kongoh can determine the number of contributors when it ranges from one to four. However, the accuracy will be affected when the number of contributors increases to reach 33% when the number of contributors becomes four. Kongoh can handle sample with a small amount of DNA, and also with degraded DNA samples. The software has a graphical user interface. R language was used to write Kongoh and its source code is available online.

On a standard desktop computer, one mixture might take around 10 h when hypothesizing 1–4 contributors. However, when hypothesizing 1–3 contributors, the runtime will decrease remarkably to a few minutes [59]. Its performance was compared to EuroForMix (version 1.7) and LRmix Studio (version 2.1.3) in [59]. In the future, authors of Kongoh are looking to use newer STR typing kits with higher sensitivity.

9.6.9 *EuroForMix*

EuroForMix is a software based on the fully continuous approach to estimate STR DNA profiles from a complex DNA sample of contributors with artifacts. It is available as an open source. EuroForMix was written in R language. Nonetheless, the likelihood function was written in C++ to speed up the computation. The software introduces a parallel implementation, since the v0.5.0, using snow R package. The parallel implementation will only be considered when a number of unknowns are at least 3 (not performed yet for database searching or non-contributor simulation). A number of processes will be similar to the number of random start points required in the optimization.

Euroformix requires a significant amount of computational time when the number of unknown contributors is four or more. Table 9.5 gives an approximation time complexity for each number of unknown contributors. From the table, it is clear that the time consumed when we have four unknown contributors was too much. Column 1 describes the number of contributors while Column 2 gives the corresponding time taken.

Table 9.5 An approximate overview of the time taken to calculate the LR depend on the number of unknown contributors [60]

Number of unknown contributors	Runtime
1	1 s
2	1 min
3	30 min
4	24 h

Table 9.6 The runtime using a different maximum number of contributors [3]

Number of contributors	Time range (Mode)
1	<1 min (0.2 min)
2	15–30 min (17 min)
3	30 min–1.5 h (1 h)
4	1–5 h (4 h)
5	5–20 h (14 h)

9.6.10 NOCIt

NOCIt [3] analyzes the DNA sample to calculate the number of contributors in a mixture. Java programming language was used to write the software. It determines the number of contributors (from 1 to 5). NOCIt can only interpret an autosomal STRs data which are independent of each other. Moreover, the software is not developed to deal with a stutter.

The execution time of [3] depends on the maximum number of contributors, the number of loci/alleles considered and the processing speed of the computer. It is also dependent on whether multiple runs of NOCIt are occurring at the same time, i.e., two NOCIt interfaces are open at once and running two separate samples. Table 9.6 provides the runtime of NOCIt. The first column gives the number of contributors, whereas the second column describes the range of time taken to analyze that number. The result was collected from a dual-core laptop with Intel® CoreTM i5-3380 CPU @ 2.9 GHz (Fig. 9.8).

9.6.11 STRmix

STRmix is a probabilistic genotyping application which performs the continuous model of interpreting the DNA profile. The DNA profile interpretation is based on a Markov Chain Monte Carlo (MCMC) sampling model [39]. It calculates the likelihood ratio which is the probability of the DNA evidence under two hypotheses, defense and prosecution hypotheses.

It was built to interpret single and mixed DNA profiles. Moreover, it follows the SWGDAM recommendations. It utilizes information that extracts from a DNA sample, such as peak height, to calculate the probability of a DNA profile for all possible genotype combinations. The software considers aspects such as allele drop-in, allele drop-out, and stutter. The software has been written in Java, and it's only available for purchase.

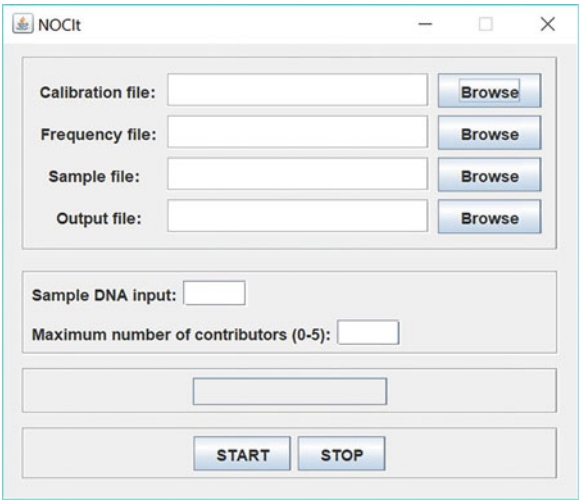


Fig. 9.8 The user interface for NOCIt v15

Moretti et al. [39] had tested STRmix and they argued that it can be used to interpret single-source profiles and mixtures of two, three, four, and five persons.

9.6.12 A Comparison of the DNA Profiling Tools

A general comparison between the selected tools is presented in Table 9.7. The first column gives the names of the software. Columns 2–8 provide information about various features of the software. Column 2 gives information on whether the software has a GUI or not. Column 3 and 4 are illustrating if the selected software considers the phenomena of drop-in and stutter on its interpretation. Column 5 describes the model that used to calculate LR. The sixth column describes the programming language that used to build the selected software. Column 7 indicates the availability of source code. The last column describes the used parallel framework. Note that the table is missing some information due to either the lack of resource for some software or because of the inability to access the software's source code.

A timeline that shows the history of introduction of the compared tools is presented in Fig. 9.9.

9.7 Conclusion

Interpreting DNA mixture is a common practice in forensic science domain. It is a complicated process that requires an extended period of time. We gave an overview of the DNA profiling field. A historical background, along with its application

Table 9.7 A general comparison between the review softwares

	GUI	Drop-in	Stutter	Calculation model	Language	Source Code	Parallelism
LRmix studio [15, 32, 50]	Yes	Yes	–	Semi-continuous	Java	Yes	Java multithreading
TrueAllele [34, 35]	Yes	Yes	Yes	Continuous	Matlab	No	–
DNAMIX V.3 [49]	Yes	–	–	–	Java	Yes	No
Euroformix [34]	Yes	Yes	Yes	Continuous	R, C++	Yes	Snow package
CeesIt [37]	Yes	Yes	Yes	Continuous	Java	No	Java multithreading
NOCIt [3]	Yes	Yes	Yes	Continuous	Java	No	Java multithreading
DNAMixtures [51]	No	Yes	Yes	Continuous	R	Yes	No
Kongoh [59, 61]	Yes	No	Yes	Continuous	R	Yes	Snow package
LikeLTD [38]	No	Yes	Yes	Continuous	R, C	Yes	OpenMP
Lab Retriever [15]	Yes	Yes	–	Semi-continuous	C++	Yes	No
STRmix [39, 56]	Yes	Yes	Yes	Continuous	Java	No	–

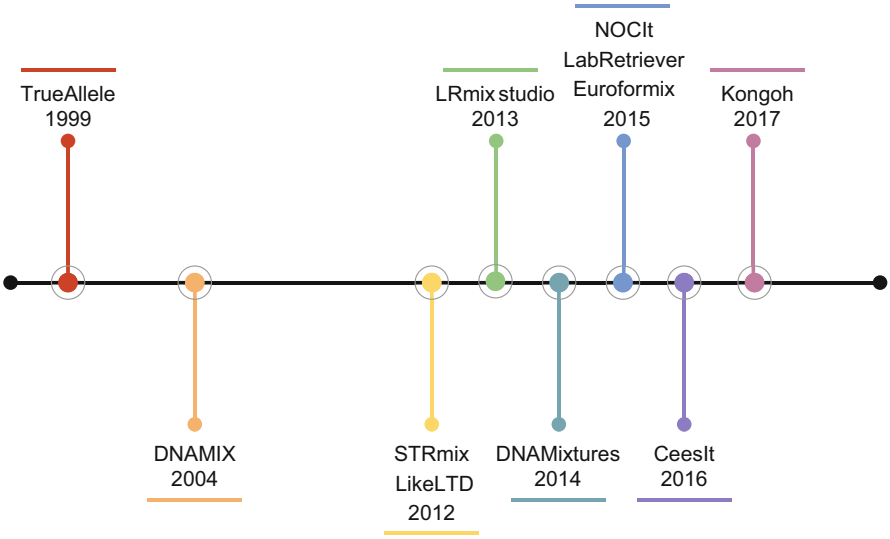


Fig. 9.9 DNA mixture analysis tools introduced over the time. This timeline describes the year of introduction of each tool

was mentioned. We, then, discuss the needed steps to sample a DNA mixture and what are the required technologies. After that, we reviewed the literature based on their classification into describing DNA profiling in general. We focus later on approaches that follow the Likelihood Ratio model. We also reviewed the various tools and compared their performance and accuracy. This is an extended version of our earlier work [4].

In the end, we would suggest the use of Euroformix and LikeLTD for DNA profiling since they are already performing parallelism. They both utilize most of the available information in the DNA sample because they follow the continuous model for calculating the LR value. The source code for the two software is available for assessment and modification. However, Euroformix provides a GUI which gives it a slight advantage over LikeLTD for users who have no technological expertise.

A frequent necessity to apply these tests might raise the need to speed up the runtime of such analysis. The computational complexity has been the major deterring factor holding the area advancements and applications. An improvement would give a chance to interpret mixtures with a larger number of unknowns and within a shorter time frame. The investigation of the relevant literature reveals that the current approaches for parallelization of DNA profiling rely on shared memory parallelization. A distributed implementation is needed to speed up the computations allowing for the use of a large number of cores and processors. This is our ongoing research, which will be reported in the near future. Faster interpretations of DNA mixtures with a large number of unknowns and higher accuracies are expected to open up new frontiers for DNA profiling in the smart societies era.

In the coming years, the complete genome sequencing technologies in a single or only a few cells will be easily available. These technologies may change the situation of DNA profiling completely. In this case, it is obvious to prepare appropriate statistical methods for that. It will be, therefore, important to prepare the mathematical and statistical algorithms for complete-genome-sequencing-based DNA profile. High-performance computing will play a key role in speeding up DNA profiling methods, particularly those HPC techniques which exploit domain-specific data and algorithmic patterns [62], system heterogeneity (e.g., disks for space, and accelerators for speed) for its advantage [63], and virtual organization models (similar to grids [64]) for information sharing across organizational boundaries. Hierarchical system structures will be needed to localize and optimize data and computations [65]. Internet of Things (IoT) would be integrated in smart city systems to create innovative services [7] and deal with big data-related challenges [6]. Mobile, fog, and cloud computing [5, 66–68] will enable dynamic system environments, seamlessly connecting users and systems.

Acknowledgments The work carried out in this chapter is supported by the HPC Center at the King Abdulaziz University.

References

1. The American Heritage medical dictionary. Houghton Mifflin Co., Boston (2007)
2. Butler, J.M.: Fundamentals of Forensic DNA Typing. Academic Press/Elsevier (2010)
3. Swaminathan, H., Grgicak, C.M., Medard, M., Lun, D.S.: NOCIt: a computational method to infer the number of contributors to DNA samples analyzed by STR genotyping. Forensic Sci. Int. Genet. **16**, 172–180 (2015)

4. Alamoudi, E., Mehmood, R., Albeshri, A., Gojobori, T.: DNA profiling methods and tools: a review. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST. pp. 216–231. Springer, Cham (2018)
5. Arfat, Y., Aqib, M., Mehmood, R., Albeshri, A., Katib, I., Albogami, N., Alzahrani, A.: Enabling smarter societies through Mobile big data fogs and clouds. Procedia Comput. Sci. **109**, 1128–1133 (2017)
6. Alam, F., Mehmood, R., Katib, I., Albogami, N.N., Albeshri, A.: Data fusion and IoT for smart ubiquitous environments: a survey. IEEE Access. **5**, 9533–9554 (2017)
7. Mehmood, R., Alam, F., Albogami, N.N., Katib, I., Albeshri, A., Altowaijri, S.M.: UTiLearn: a personalised ubiquitous teaching and learning system for smart societies. IEEE Access. **5**, 2615–2635 (2017)
8. Butler, J.M.: The future of forensic DNA analysis. Philos. Trans. R. Soc. Lond. Ser. B Biol. Sci. **370**, 577–579 (2015)
9. Paoletti, D.R., Krane, D.E., Raymer, M.L., Doom, T.E.: Inferring the number of contributors to mixed DNA profiles. IEEE/ACM Trans. Comput. Biol. Bioinforma. **9**, 113–122 (2012)
10. Perez, J., Mitchell, A.A., Ducasse, N., Tamariz, J., Caragine, T.: Estimating the number of contributors to two-, three-, and four-person mixtures containing DNA in high template and low template amounts. Croat. Med. J. **52**, 314–326 (2011)
11. Gill, P., Haned, H.: A new methodological framework to interpret complex DNA profiles using likelihood ratios. Forensic Sci. Int. Genet. **7**, 251–263 (2013)
12. Weedn, V.W., Foran, D.R.: Forensic DNA typing. In: Molecular pathology in clinical practice. pp. 793–810. Springer International Publishing, Champions (2016)
13. Monich, U.J., Grgicak, C., Cadambe, V., Wu, J.Y., Wellner, G., Duffy, K., Medard, M.: A signal model for forensic DNA mixtures. In: 2014 48th Asilomar Conference on Signals, Systems and Computers. pp. 429–433. IEEE (2014)
14. Tao, R., Wang, S., Zhang, J., Zhang, J., Yang, Z., Sheng, X., Hou, Y., Zhang, S., Li, C.: Separation/extraction, detection, and interpretation of DNA mixtures in forensic science (review)
15. Inman, K., Rudin, N., Cheng, K., Robinson, C., Kirschner, A., Inman-Semerau, L., Lohmueller, K.E.: Lab retriever: a software tool for calculating likelihood ratios incorporating a probability of drop-out for forensic DNA profiles. BMC Bioinformatics. **16**, 298 (2015)
16. Schmidt, B., Hildebrandt, A.: Next-generation sequencing: big data meets high performance computing. Drug Discov. Today. **22**, 712–717 (2017)
17. Chang, Y.-J., Chen, C.-C., Chen, C.-L., Ho, J.-M.: A de novo next generation genomic sequence assembler based on string graph and MapReduce cloud computing framework. BMC Genomics. 13 Suppl 7, S28 (2012)
18. Li, D., Liu, C.-M., Luo, R., Sadakane, K., Lam, T.-W.: MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. Bioinformatics. **31**, 1674–1676 (2015)
19. Liu, Y., Schmidt, B., Maskell, D.L.: DecGPU: distributed error correction on massively parallel graphics processing units using CUDA and MPI. BMC Bioinformatics. **12**, 85 (2011)
20. Erbert, M., Rechner, S., Müller-Hannemann, M.: Gerbil: a fast and memory-efficient k-mer counter with GPU-support. Algorithms Mol. Biol. **12**, 9 (2017)
21. Varma, B.S.C., Paul, K., Balakrishnan, M., Lavenier, D.: FAssem: FPGA Based Acceleration of De Novo Genome Assembly. In: 2013 IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines. pp. 173–176. IEEE (2013)
22. Ramachandran, A., Heo, Y., Hwu, W.M., Ma, J., Chen, D.: FPGA accelerated DNA error correction, https://iwe.pure.elsevier.com/en/publications/fpga-accelerated-dna-error-correction, (2015)
23. Kang, S.J., Lee, S.Y., Lee, K.M.: Performance comparison of OpenMP, MPI, and MapReduce in practical problems. Adv. Multimed. **2015**, 1–9 (2015)
24. Hamidi, B., Hamidi, L.: Synchronization Possibilities and Features in Java, vol. 1, p. 75 (2015)
25. Carpenter, B., Getov, V., Judd, G., Skjellum, A., Fox, G.: MPJ: MPI-like message passing for Java. Concurr. Pract. Exp. **12**, 1019–1038 (2000)

26. Memeti, S., Pllana, S.: A machine learning approach for accelerating DNA sequence analysis. Int. J. High Perform. Comput. Appl. 1–17
27. Bell, G., Gray, J.: What' S Next in Computing ? 45, 91–95 (2002)
28. Diegoli, T.M., Rohde, H., Borowski, S., Krawczak, M., Coble, M.D., Nothnagel, M.: Genetic mapping of 15 human X chromosomal forensic short tandem repeat (STR) loci by means of multi-core parallelization. Forensic Sci. Int. Genet. **25**, 39 (2016)
29. Laguna, I., Ahn, D.H., De Supinski, B.R., Gamblin, T., Lee, G.L., Schulz, M., Bagchi, S., Kulkarni, M., Zhou, B., Chen, Z., Qin, F.: Debugging high-performance computing applications at massive scales. Commun. ACM. **58**, 72–81 (2015)
30. Butler, J.M.: Advanced topics in forensic DNA typing: interpretation
31. Bille, T., Bright, J.-A., Buckleton, J.: Application of random match probability calculations to mixed STR profiles. J. Forensic Sci. **58**, 474–485 (2013)
32. Garofano, P., Caneparo, D., D'Amico, G., Vincenti, M., Alladio, E.: An alternative application of the consensus method to DNA typing interpretation for low template-DNA mixtures. Forensic Sci. Int. Genet. Suppl. Ser. **5**, e422–e424 (2015)
33. Kelly, H., Bright, J.-A., Buckleton, J.S., Curran, J.M.: A comparison of statistical models for the analysis of complex forensic DNA profiles. Sci. Justice. **54**, 66–70 (2014)
34. Bleka, Ø., Storvik, G., Gill, P.: EuroForMix: an open source software based on a continuous model to evaluate STR DNA profiles from a mixture of contributors with artefacts. Forensic Sci. Int. Genet. **21**, 35 (2016)
35. Perlin, M.W., Dormer, K., Hornyak, J., Schiermeier-Wood, L., Greenspoon, S.: TrueAllele casework on Virginia DNA mixture evidence: computer and manual interpretation in 72 reported criminal cases. PLoS One. **9**, e92837 (2014)
36. Gill, P., Haned, H., Eduardoff, M., Santos, C., Phillips, C., Parson, W.: The Open-source software LRmix can be used to analyse SNP mixtures. Forensic Sci. Int. Genet. Suppl. Ser. **5**, e50 (2015)
37. Swaminathan, H., Garg, A., Grgicak, C.M., Medard, M., Lun, D.S.: CEESIt: a computational tool for the interpretation of STR mixtures. Forensic Sci. Int. Genet. **22**, 149–160 (2016)
38. Balding, D.J., Steele, C.: The likeLTD software: an illustrative analysis, explanation of the model, results of performance tests and version history. UCL Genet. Inst. **1**, 1–49 (2014)
39. Moretti, T.R., Just, R.S., Kehl, S.C., Willis, L.E., Buckleton, J.S., Bright, J.-A., Taylor, D.A., Onorato, A.J.: Internal validation of STRmix™ for the interpretation of single source and mixed DNA profiles. Forensic Sci. Int. Genet. **29**, 126–144 (2017)
40. Taylor, D., Bright, J.-A., Buckleton, J.: Interpreting forensic DNA profiling evidence without specifying the number of contributors. Forensic Sci. Int. Genet. **13**, 269–280 (2014)
41. Russell, D., Christensen, W., Lindsey, T.: A simple unconstrained semi-continuous model for calculating likelihood ratios for complex DNA mixtures. Forensic Sci. Int. Genet. Suppl. Ser. **5**, e37–e38 (2015)
42. Paoletti, D.R., Doom, T.E., Krane, C.M., Raymer, M.L., Krane, D.E.: Empirical analysis of the STR profiles resulting from conceptual mixtures. J. Forensic Sci. **50**, JFS2004475–JFS2004476 (2005)
43. Biedermann, A., Bozza, S., Konis, K., Taroni, F.: Inference about the number of contributors to a DNA mixture: comparative analyses of a Bayesian network approach and the maximum allele count method. Forensic Sci. Int. Genet. **6**, 689–696 (2012)
44. Haned, H., Pène, L., Sauvage, F., Pontier, D.: The predictive value of the maximum likelihood estimator of the number of contributors to a DNA mixture. Forensic Sci. Int. Genet. **5**, 281–284 (2011)
45. Haned, H., Pène, L., Lobry, J.R., Dufour, A.B., Pontier, D.: Estimating the number of contributors to forensic DNA mixtures: does maximum likelihood perform better than maximum allele count? J. Forensic Sci. **56**, 23–28 (2011)
46. Haned, H., Benschop, C.C.G., Gill, P.D., Sijen, T.: Complex DNA mixture analysis in a forensic context: evaluating the probative value using a likelihood ratio model. Forensic Sci. Int. Genet. **16**, 17–25 (2015)

47. Egeland, T., Dalen, I., Mostad, P.F.: Estimating the number of contributors to a DNA profile. Int. J. Legal Med. **117**, 271–275 (2003)
48. Marciano, M.A., Adelman, J.D.: PACE: probabilistic assessment for contributor estimation—a machine learning-based assessment of the number of contributors in DNA mixtures. Forensic Sci. Int. Genet. **27**, 82–91 (2017)
49. Curran, J.M., Triggs, C.M., Buckleton, J., Weir, B.S.: Interpreting DNA mixtures in structured populations. J. Forensic Sci. **44**, 987–995 (1999)
50. Haned, H., De Jong, J.: LRmix Studio 2.1 user manual. (2016)
51. Graversen, T.: Statistical and Computational Methodology for the Analysis of Forensic DNA Mixtures with Artefacts, https://ora.ox.ac.uk/objects/uuid:4c3bfc88-25e7-4c5b-968f-10a35f5b82b0, (2014)
52. Forensim: An open-source initiative for the evaluation of statistical methods in forensic genetics. Forensic Sci. Int. Genet. 5, 265–268 (2011)
53. Gill, P., Sparkes, R., Pinchin, R., Clayton, T., Whitaker, J., Buckleton, J.: Interpreting simple STR mixtures using allele peak areas. Forensic Sci. Int. **91**, 41–53 (1998)
54. Kling, D., Egeland, T., Tillmar, A.O.: FamLink – a user friendly software for linkage calculations in family genetics. Forensic Sci. Int. Genet. **6**, 616–620 (2012)
55. Tvedebrink, T., Eriksen, P.S., Mogensen, H.S., Morling, N.: Evaluating the weight of evidence by using quantitative short tandem repeat data in DNA mixtures. J. R. Stat. Soc. Ser. C Applied Stat. **59**, 855–874 (2010)
56. Developmental validation of STRmix™, expert software for the interpretation of forensic DNA profiles. Forensic Sci. Int. Genet. 23, 226–239 (2016)
57. Perlin, M.W., Hornyak, J.M., Sugimoto, G., Miller, K.W.: TrueAllele genotype identification on DNA mixtures containing up to five unknown contributors*, vol. 60, p. 857 (2015)
58. Cowell, R.G., Graversen, T., Lauritzen, S.L., Mortera, J.: Analysis of forensic DNA mixtures with artefacts. J. R. Stat. Soc. Ser. C Applied Stat., 64. 1–48 (2015)
59. Manabe, S., Morimoto, C., Hamano, Y., Fujimoto, S., Tamaki, K.: Development and validation of open-source software for DNA mixture interpretation based on a quantitative continuous model. PLoS One. **12**, e0188183 (2017)
60. Bleka, Ø.: An introduction to EuroForMix (v1.8). 2016, 1–59 (2016)
61. Manabe, S.: Kongoh version 1.0.1 User Manual. 1–12 (2017)
62. Mehmood, R., Crowcroft, J.: Parallel iterative solution method for large sparse linear equation systems. Comput. Lab. Univ. 22 (2005)
63. Mehmood, R.: Serial disk-based analysis of large stochastic models. In: Validation of Stochastic Systems. pp. 230–255. Springer, Berlin, (2004)
64. Altowaijri, S., Mehmood, R., Williams, J.: A quantitative model of grid systems performance in healthcare organisations. In: 2010 International Conference on Intelligent Systems, Modelling and Simulation. pp. 431–436. IEEE (2010)
65. Mehmood, R., Crowcroft, J., Hand, S., Smith, S.: Grid-level computing needs pervasive debugging. In: The 6th IEEE/ACM International Workshop on Grid Computing, 2005. p. 8 pp. IEEE (2005)
66. Tawalbeh, L.A., Mehmood, R., Benkhlifa, E., Song, H.: Mobile cloud computing model and big data analysis for healthcare applications. IEEE Access. **4**, 6171–6180 (2016)
67. Tawalbeh, L.A., Bakhader, W., Mehmood, R., Song, H.: Cloudlet-Based Mobile Cloud Computing for Healthcare Applications. In: 2016 IEEE Global Communications Conference (GLOBECOM). pp. 1–6. IEEE (2016)
68. Muhammed, T., Mehmood, R., Albeshri, A., Katib, I.: UbeHealth: A personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities, https://ieeexplore.ieee.org/document/8382164/, (2018)

Chapter 10
An Architecture to Improve the Security of Cloud Computing in the Healthcare Sector

Saleh M. Altowaijri

10.1 Introduction

Cloud technology is a widely adopted technology in the present era. This technology has given new life to all business organizations. It is also used in the healthcare sector and is increasing business flexibility in medical organizations. Flexibility, pay-as-you-go, cost effectiveness, greater efficiency, and agility are some of the benefits of this technology. While there are many advantages, there are also some risks, particularly with regard to the security of data in the cloud, which is the most challenging issue at all times. In cloud computing this has became more problematic because the actual data are stored in another location. So, provision of security for the data in the cloud is a tedious task for cloud computing organizations. We are talking here only about the healthcare cloud.

At present the healthcare sector requires creation of an environment that reduces time-consuming efforts and other costly operations to obtain a patient's complete medical records and uniformly integrate this heterogeneous collection of medical data to deliver them to the healthcare system. Electronic health records (EHRs) have been widely adopted to enable healthcare providers and patients to create, manage, and access healthcare information from any place and at any time. Cloud services provide the necessary infrastructure at lower cost and better quality. Cloud computing, when used in the healthcare sector, reduces the cost of storing, processing, and updating, with improved efficiency and quality. But the security of data in the cloud is not satisfactory today. The EHR consists of images of the patient's records, which are highly confidential. EHRs in healthcare include scan

S. M. Altowaijri (✉)
Faculty of Computing and Information Technology, Northern Border University, Rafha, Kingdom of Saudi Arabia
e-mail: Saltowaijri@nbu.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*, EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_10

images, DNA reports, x-rays, etc., which are considered the patient's private data. Provision of security for a large volume of data with high efficiency is required. Data in the healthcare cloud are in an encrypted form. These data are very important and an attractive target for cybercriminals. Many researchers have proposed architecture to secure the healthcare cloud, and many techniques for securing the data in the cloud have been investigated. These researchers are both industry experts and academicians. Here, we present some of the researchers' previous work.

Kim et al. have presented a trusted model for efficient reconfiguration and allocation of computing resources, depending upon the user's request [1]. Trust calculations are made to achieve reliability. A collaborative trust model of firewall-through based on cloud computing has been proposed by Yang et al. [2]. A protocol to establish trust and confidentiality while accessing data has been proposed by Ahmed et al. [3]. Brodkin [4] has recognized seven security risks that are essential to consider before enterprises make decisions regarding transformation into a cloud computing model. Cloud computing as an approach introduces new risks, influences others, and magnifies some, according to Chen and Zhao [5]. These risks and their effect on security risks and vulnerabilities have been explained by Grobauer [6].

In earlier work, Mehmood and colleagues looked at the use of grid and cloud computing in healthcare [7, 8], transport [9–11], and distance learning [12]. In this chapter, we discuss security issues in the healthcare cloud and propose architecture to secure data in the healthcare cloud.

Section 10.2 of the chapter gives an overview of cloud computing, cloud architecture, and the advantages of cloud computing. Section 10.3 discusses the great benefits that the use of cloud computing can bring to healthcare organizations. Sections 10.4, 10.5, and 10.6 discuss cloud computing security, methods of cloud security, and security threats in the healthcare cloud, respectively. Section 10.7 describes the background to secure healthcare cloud architecture and reviews the relevant literature. Section 10.8 introduces the proposed secured architecture for the healthcare cloud and the results of using it. Section 10.9 concludes the chapter.

10.2 Cloud Computing: An Overview

Cloud computing is the spread of computing services such as servers, storage, databases, networking, software, machines and more devices over the internet, which is known as "the cloud." Those organizations who offer these services are called cloud providers and normally request money for cloud computing services on the basis of their usage, similarly to how electricity or water are paid for at home. Figure 10.1 illustrates a typical cloud, which is accessed through various devices and infrastructure.

Fig. 10.1 Cloud architecture

10.2.1 Types of Cloud Services: IaaS, PaaS, and SaaS

Cloud computing services are divided into three categories: infrastructure-as-a-service (IaaS), platform-as-a-service (PaaS), and software-as-a-service (SaaS). These are like a computing stack, because they are created on top of one another and look like a stack. The following is a brief introduction to each type. Figure 10.2 depicts these three service categories.

(a) *Infrastructure-as-a-service* (IaaS): This is the most basic category of cloud computing. By IaaS, we mean the information technology (IT) infrastructure such as servers, virtual machines (VMs), storage, networks, operating systems, etc., from a cloud service provider on a pay-as-you-go basis [13].
(b) *Platform-as-a-service* (PaaS): This refers to cloud computing services that provide an on-demand environment for developing, testing, delivering, and managing software built applications. PaaS is considered to make it easier for developers to rapidly develop web or mobile apps, without thinking about managing the original infrastructure of servers, storage, networks, and VMs needed for development.
(c) *Software-as-a-service* (SaaS): This is a technique for providing software applications over the internet, on demand and typically on a pay-as-you-go basis. In using SaaS, cloud providers host the infrastructure and platform by using the internet, which can be connected to by using web browsers.

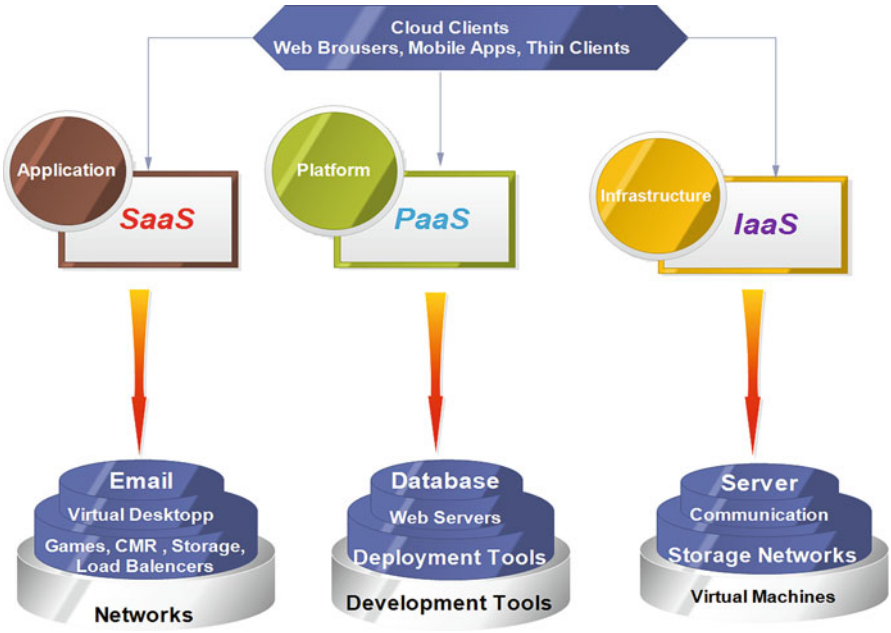


Fig. 10.2 Cloud computing overview

10.2.2 Advantages of Cloud Computing

The healthcare sector is switching to cloud computing instead of traditional IT solutions. Its main reasons are to manage dynamic needs for computational resources, scalability of human resources, high infrastructure management costs, and increases in demands for collaboration, multitenancy, and ubiquitous access. To overcome all of these issues, cloud computing offers the simplest and best solutions with cost effectiveness. These solutions are on-demand services, broad network access, resource pooling, measured service, and elasticity. These services are favorable, and their costs and maintenance requirements are easy for both clients and organizations to bear [14, 15].

Hence, cloud computing is a big move to uplift anyone's business. Let us think about IT resources when there was no cloud computing, so we can easily understand benefits of cloud computing. Why is cloud computing so popular?

Here, we give some common answers to these questions, by which we can easily understand why organizations are turning to cloud computing services. The reasons are:

1. *Cost effectiveness*: Cloud computing reduces the principal costs of buying hardware and software, and the costs of managing and running an on-site data center, i.e., a clusters of servers, round-the-clock electricity for light and cooling, IT personnel for setting up the infrastructure, and many more costs.

2. *Velocity*: Most cloud computing services provide personal service on demand, so that even large volumes of data can be provisioned in seconds, generally in just a few clicks, giving businesses a lot of ease and taking the stress off capability planning [13].
3. *Global scaling* (*regions*): The advantages of cloud computing services include the ability to scale elastically. In the cloud, that means providing the required IT resources such as servers, computing control database, storage and networking when they are needed and from the right regional location.
4. *Enhanced productivity*: On-site data centers typically require a lot of "racking and stack" hardware setup, software patching, and other sustained IT management tasks. Cloud computing removes the requirement for many of these odd jobs so that IT teams can use their time to achieve more significant business objectives [13].
5. *Better performance*: The biggest cloud computing services run on a universal network of protected data centers, which are frequently upgraded to the latest generation of fast and well-organized computing hardware. This offer several advantages over a single business data center, including compact network latency for application and greater economy of scale.

10.3 Cloud Computing and Healthcare

Trends in healthcare organizations have major impacts on health IT systems. There is a huge escalation in demand for healthcare services because of population increases and the increasing prevalence rates of chronic diseases. Moreover, there are capital pressures stemming from the requirement to do extra work and good-quality work with lesser and more costly resources and also reduced income. Expectations for improved results, good-quality treatment, and more value from the healthcare services that are provided raise the requirements for point-of-care access to medical data, and parallel evolution and adoption of mobile devices, by both medical staff and patients, are increasing the need for IT systems to become customized. Also, the major increase in digitization of health records—including greater acceptance of electronic medical records (EMRs), electronic health records (EHRs), and personal health records (PHRs), and the growing frequency of digital outputs from scanning and monitoring machines, such as magnetic resonance imaging (MRI) scanners and bedside monitors and infusers—provide more capacious and mixed digital data to take advantage of the possible advantages of cloud solutions. Healthcare provider systems deploying cloud-based computing and cloud services reap various benefits in contrast to those using domestic client–server systems, including financial, and functional advantages. The financial profit of cloud computing can be major, since cloud computing offers cost flexibility and the possibility of cost savings. Heavy asset expenses can be avoided because IT assets are acquired on demand as needed and paid for as operating expenditure. Also, the cost of the workforce required to organize and

maintain IT assets is built into the cost of cloud computing, so the need for further healthcare supplier–trained IT staff and the associated costs may be decreased when cloud services are used for IaaS and PaaS platforms, and even more so for SaaS solutions, where the cloud service provider takes on the major share of the work. From an operational viewpoint, cloud services offer elasticity and the ability to adjust to demand quickly. Cloud services can propose better security and privacy for health data and health systems. Cloud service provider data centers are normally very safe and well secured against stranger and insider threats by use of administrative, physical, and technical methods implemented and maintained by expert professional staff. Cloud services can offer sophisticated security controls, including data encryption and fine-grained access controls and access logging. Medical systems created by using cloud services can give web access to information, avoiding the necessity to save information on consumer devices. The requirement for limited IT security skills within the healthcare sector is also minimized. Cloud service providers normally function on such a level that they have all of the required IT skills, with the range of those skills being spread across many customers. Healthcare functionality can be improved by cloud-based healthcare IT systems that propose the possibility of broad interoperability and integration. Healthcare cloud services are internet based and usually use normal protocols; thus, connecting them to other systems and applications is typically simple, although EHR/EMR vendor contractual and scientific impediments continue to present a challenge. The key to sharing information simply and securely is complex potential, and cloud services are good enablers for this. Cloud services also maintain fast progress and improvements, particularly for mobile and internet of things (IoT) devices, thus meeting the demands imposed on healthcare IT systems by these new and rapidly advancing technologies. Cloud services can enable remote ways into applications and data via the internet through use of wired and wireless systems to enable access at any time from anywhere that internet connectivity can be established. Also, cloud services present the right to use to a much enhanced ecosystem of healthcare suppliers, financiers, life science entities, and IT solution buddies, all of which raise the potential for a wide variety of services to healthcare provider industries. The main difference between traditional IT and cloud services is the way of sharing responsibilities. In traditional IT, the IT organization is responsible for almost everything. With cloud services, responsibility is shared between the cloud service provider and the healthcare organization as the cloud service customer [16]. Perhaps the greatest functional advantage of healthcare cloud services is the wide range of new capabilities that they are able to propose. These services offer the chance to extend the capability on hand to health organization employees, in order to apply better ways of working and to offer new services to patients. Complicated analytical capability can be brought to bear to achieve better patient-specific and population-based appraisal and organization [17].

10.4 Cloud Computing Security

Nowadays, people are very conscious about their health; this is also the biggest business in the world. People can pay a lot of money to doctors and hospitals to save their lives. From the business point of view, this is a business whose demise will never occur. Before the availability of technology, the hospital was the only medium for provision of healthcare, but nowadays the scene has changed. Most people have adopted these services as a business, and healthcare is now provided online. This has become possible only because of cloud computing. With the help of cloud computing, companies are changing their ways of providing services, e.g., by offering online consultations with doctors or online clinics and pharmacies, with impacts on the quality of service delivery and the cost of these services. To manage these changes, two forces are applied: the first is to fulfill the business imperative to cut costs, and the second is to improve the quality of healthcare services. In the past 10 years, a large number of hospital IT departments have started to use good backup and disaster recovery (DR) tools to keep their systems and data safe and recoverable in the event of a system failure. Hospital users have always been assured that their IT staff can promise a system uptime of 99.9%. However, with the increasing use of cloud services for data protection purposes, IT must adjust to the new reality of cloud-based DR options. For this, they use DRaaS (disaster-recovery-as-a-service) [13].

The appearance of cloud computing technology with major advantages is one of the present key challenges. This is a new prototyping technology based on "pay-on-demand" for the use of information and communications technology (ICT) [18]. The National Institute of Standards and Technology (NIST) in the USA has focused on three models of cloud computing: SaaS, PaaS, and IaaS [19]. In healthcare cloud computing for internal communications, an extensive number of computers and servers are dedicated to meeting the requirements of the medical care business. Healthcare services can be delivered to users (patients or physicians) through an internet connection [20].

First, there is SaaS, where the cloud service provider provides access to particular software functions, such as table processing or email. The cloud service provider also manages any software upgrades and fixes protection problems. In PaaS, clients may have remotely accessible computing control and can run their personal applications. However, maintenance is the responsibility of the cloud providers. Finally, IaaS is a latent option. In this scenario, customers may have remotely accessible computing control, are able to run some of their own applications, and are charged for resolution of any maintenance problems. There are many advantages to using the healthcare cloud, such as allowing enclosed entities to store information off-site. Moreover, if employees need to work remotely or move from one location to another, healthcare cloud options provide them with the liberty to do so while still being able to access important and critical information [21, 22].

Additionally, this can assist organizations to reduce their operating or storage costs, update services, and devote more resources to maintenance of software, platforms, or infrastructure.

Also, it is important to note that the [US] Health Insurance Portability and Accountability Act (HIPAA) compilation rule requires patient data to be well protected, regardless of where it is stored. Organizations that are working as contractor firm and do not necessarily analyze the data on a normal basis must adhere to HIPAA rules. This particular system records every access attempt by the username and include the date, time, relationship to the patient, etc. Still, more research work is required in this field to increase the security of patient data and users' trust levels [16, 23].

Cloud computing has some major security issues. Because they have only limited cybersecurity resources, many healthcare service providers have become vulnerable to various attacks and have attracted cybercriminals [24]. Cloud computing has a similar name to internet computing. How safe are our data? Data security is the biggest concern in cloud computing. Reliability, authentication, availability, and integrity are different aspects of data security. Reliability is related to trust in computing. How we can trust cloud computing when we are not there? A person should not share his or her data over the cloud if he or she is not comfortable with the internet. Besides reliability there are many other security concerns in cloud computing, such as authenticity, data locality, licensing security, and physical damage. In the next section we discuss some of these security issues with their previously proposed solutions. Here, we define the major security concerns for any type of computing [25].

1. *Authentication*: Authentication is the process of confirming the truth about an entity or a piece of data. Authenticity is a phenomenon that allows users to use particular services.
2. *Confidentiality*: Confidentiality means a set of rules that restrict access to some information to certain individuals.
3. *Integrity*: Integrity in terms of cloud security is the assurance that only authorized or authenticated users can access or modify the data.
4. *Availability*: Availability, in the context of a computer system, means the ability to access data, information, or resources in an appropriate format. It must be ensured by the storage, which may be local or at an off-site facility.
5. *Nonrepudiation*: This means that neither the sender nor the receiver can deny the validity of the data or information.

The above are the major security concerns in any type of computing. All issues related to cloud computing refer to one of the above security concerns. In cloud computing there are three components: SaaS, PaaS, and IaaS.

10.5 Methods of Cloud Security

On one side, the job of cybercriminals is to steal confidential data. On the other side, researchers and security experts propose the architecture needed to make data secure in the cloud.

Multitenant Platform This healthcare cloud platform has been published in a paper by Oh et al. [13]. This healthcare SaaS platform (HSP) provides an easy-to-use, cloud-based, modular EHR system. In this architecture, the functional and software analysis of an HSP has been designed in a layered architecture. Exterior systems can interface with the HSP by using the Simple Object Access Protocol (SOAP) and Representational State Transfer/JavaScript Object Notation (REST/JSON). The multitenancy model of the HSP is designed as a shared database, with a different schema for each tenant through a single application, although healthcare data can be physically located in the cloud or at a hospital, depending on regulations. The Consumer Directed Services (CDS) services are categorized into rule-based services for medications, alert registration services, and knowledge services. The above process of multitenant architecture is depicted in Fig. 10.3.

How to protect the data Protection of critical patient information and medical records is one of the most basic duties of the healthcare industry and one of the most firmly regulated. To defend data as they move in and out of the cloud requires data

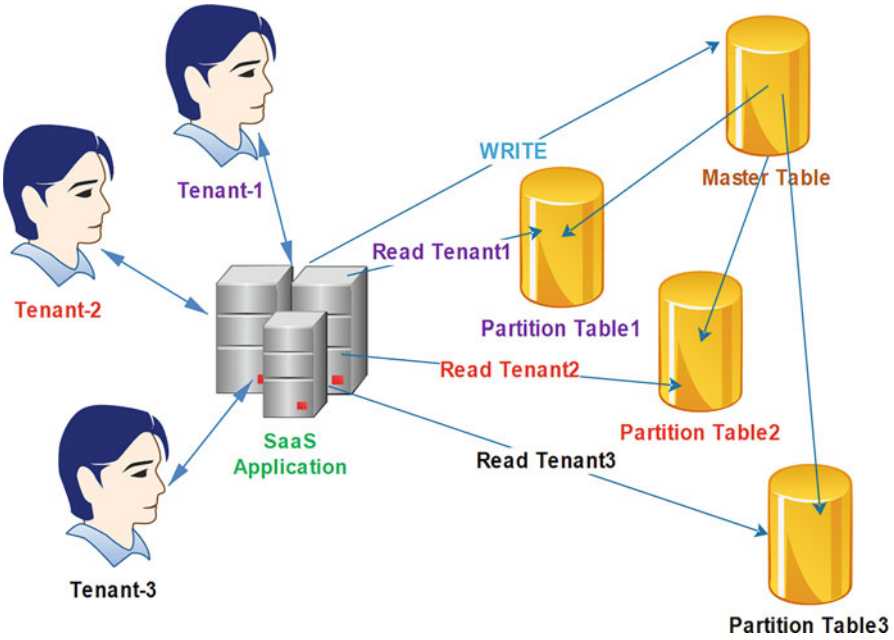


Fig. 10.3 Multitenant storage model

encryption, which makes the data unusable if they are compromised. It also demands safe communication connections, which limit browser access and encrypt content as it is moved over the network and throughout the cloud. However, data encryption based on the Advanced Encryption Standard (AES) algorithm is very compute intensive. This type of software-based encryption relies on compute-intensive algorithms that can impact the performance of the computing network, particularly when used pervasively to protect the massive volumes of information that pass to and from the cloud. Traditional encryption solutions can create computing logjams due to high performance overheads, making them less than optimal for protecting cloud data traffic. Intel has worked to mitigate these performance penalties [26].

How to provide security against unauthorized access Realizing cloud computing advantages while meeting stringent requirements for data security and compliance requires hardening of the underlying platform, including the hardware, software, and process methodologies. Better securing of both server and client platforms helps safeguard cloud infrastructures, and better management of identities and access control points at the network edges helps ensure that only authorized users can enter the cloud. With malware attacks now moving beyond software to target the platform, organizations face new risks from rootkit and other low-level exploits that can infect system components such as hypervisors and the BIOS to quickly spread throughout the cloud environment.

Protection of identity in the cloud Protection of identity on a cloud platform begins with managing who has access to it. Identity protection devices (such as Intel® IPT) provide a simple way for healthcare organizations to validate that legitimate employees or approved users are allowed in from a trusted device. IPT offers token generation incorporated into the hardware, which gets rid of the need for (and cost of) a different physical token. It also confirm transactions and protects against malware [27]. Figure 10.4 explains the extra security layer in the healthcare cloud. Any user who wants to access a cloud application first needs to enter his or her credentials (username and password) on the identity protection system and then receives a one-time password (OTP) on his or her registered cell phone or email address. Only if both are correct will the identity protection system allow that user to access the cloud.

Protection of API keys Application programming interfaces (APIs) are the fundamental method used for exposing cloud applications to third parties and mobile services. A hacker tries to break these API keys for unauthorized access. Many researchers and scientists have suggested algorithms to protect API keys [28].

10.6 Security Threats in the Healthcare Cloud

Healthcare organizations have always struggled with information security. Because the healthcare industry stores massive volumes of critical data and is subject to strict compliance rules, it must make security its primary concern. Therefore, the

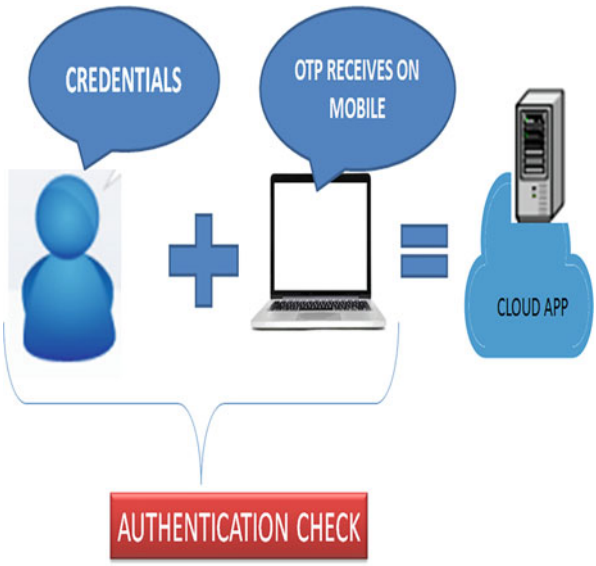


Fig. 10.4 Identify protection in the cloud

industry has long been doubtful about new technologies that could put data at risk, including cloud technologies. Cloud computing poses many risks to data security, data confidentiality, and overheads because of the huge volumes of data involved. Data processed in the cloud are highly confidential, such as business records, patient records, military records, etc. Therefore, proper encryption standards and architecture must be applied to secure sensitive data against tampering [29].

However, everything changes, and the healthcare industry is changing as well. In January 2018, an important decision was made: the National Health Service (NHS)—the largest healthcare provider in the UK—officially approved the use of US-based cloud providers to store patient data. According to the *2018 Netwrix Cloud Security In-Depth Report*, 84% of healthcare organizations already store data in the cloud, but the NHS is the first state healthcare organization to give the go ahead [4, 22, 29].

Here, we discuss some of the major security risks in the healthcare cloud.

Malware and viruses Malware and viruses are being developed continuously, and ransomware (a type of malware that, once it has taken over the computer, threatens harm) is one of the most frequent sources of attack. According to one report, a company is targeted by ransomware every 40 seconds. Malware—such as NotPetya, WannaCry, and Locky, in particular—has spread among healthcare providers. Even the NHS itself has been targeted by WannaCry: the attack resulted in disruptions at 37% of NHS organizations and cancellation of many appointments and surgeries. Although the NHS did not pay the ransom, it did incur extra costs to cover cancelled appointments, hire IT consultants, and restore data and systems after the attack, besides incurring damage to its reputation. Unsurprisingly, nearly 61% of healthcare organizations are reportedly worried about malware and the threat of unauthorized access [30].

Identity protection and access management Unauthorized access is the biggest challenge in all types of cloud computing. This is a major security issue throughout the world and a huge challenge in healthcare cloud computing. Many researchers and IT industry developers are working to resolve this issue. According to a Netwrix survey in January 2018, 68% of unauthorized access security concerns are related to the healthcare cloud. This is the biggest security issue. Existing organizational identification and authentication frameworks may not expand into the cloud, and if these are based on unique username–password combinations for individual applications, they can be a weak link in the security chain. In the cloud, identity management helps to preserve security, visibility, and management, and centralization of IT control of identities and access is useful.

Data encryption Data saved in the cloud usually reside in a multitenant environment—a distribution virtualized server space—with data from other clients of the cloud provider. Healthcare entities that move critical and synchronized data into the cloud must make sure the data are encrypted at rest and in transit. One of the main risks of multitenancy and shared computing resources within cloud infrastructures is possible failure of the separation instrument that provides separation of memory, storage, and routing between tenants.

Data compliance regulations Security laws and regulations vary at national, regional, and local levels, making fulfillment a potentially complex issue for cloud computing. For example, some countries in the European Union (EU) stipulate that some health data must never cross those countries' own borders. Other authorities have detailed data compliance regulations that stipulate special handling of certain kinds of health information (medical treatment of minors, disease history, etc.), controlling transmission across local or state borders. To comply with these strict data privacy laws, cloud infrastructures must be auditable for such features as encryption, security controls, and geometric location.

Illegal activities of IT staff Although it seems strange, employees have been identified as a security threat. Only 21% of healthcare industries have a complete perception of what their IT staff members are doing in the cloud, and visibility of the actions of business users is even rarer. Actually, the overall visibility of inner actors is the lowest among all organizations surveyed. IT people are aware of this difference, but the majority of them do not get essential support from the C-level to address it. Only 50% of respondents say that they get top management support to implement cloud security projects; this is the lowest outcome across all businesses surveyed.

Human error This is also one of the biggest security threats; with just one small mistake, the industry can lose billions of dollars within a second. According to Verizon's *2016 Breach Investigations Report*, healthcare data breaches in 2015 were most likely to be caused by human error or unintentional error in the form of stolen or lost assets, insider and privilege misuse, and miscellaneous errors, such as improper device disposal or mishandling.

Detailed above are some of the common threats that are spreading in the healthcare cloud. The healthcare cloud also contains massive volumes of data. Thus, the healthcare industry is worried about protecting these data. The HIPAA and public health authorities (PHAs) have issued regulations to secure data in the healthcare cloud. In the next section we describe some methods by which the healthcare industry can save its data in the cloud.

10.7 Secure Healthcare Cloud Architecture

From past studies it has become clear that a large number of cybercriminals are targeting the healthcare cloud. The reason behind this is that it is the most crucial cloud and can generate a terabyte of data in a single day. Also, people are less vigilant about security of health information than about the security of banking or other organizations' information, so this cloud is the easiest target for hackers. However, in recent years, researchers have worked to ensure that data in the healthcare cloud are censured and have proposed some architecture. Some of these architectures are explained in this chapter.

The architecture proposed by Chondamrongkul and Chondamrongkul is very similar to our method. This supports a healthcare system that allow patients to be checked by mobile applications. A personal record application helps gather health data from secure mobile cloud architecture for linked wearable devices and cell phones, before saving them in the cloud. After that, a monitoring application retrieves these data to enable doctors and other relevant medical staff to supervise the patient's condition [31].

Zhang and Liu have presented a paper in which they discuss EHR sharing and integration in healthcare and analyze arising security and privacy issues in access to and management of EHRs [14].

In 2013, Khan and Wan proposed an architecture to make data secure in the healthcare cloud. In this work they introduced a trusted authority between the cloud and the user. When any user want to access the healthcare cloud, it must be passed by the trusted authority, which is usually linked to the private key store. They gave a review of the wireless body area network (WBAN). They provided an outlook on this promising field and discussed a cloud-enabled WBAN architecture for pervasive healthcare systems. This system can be accessed by smart phones with an enabled Wi-Fi connection or something similar [20].

10.8 Our Architecture

We have obtained help from big data to solve this issue. In our architecture to store the data, there are some slave nodes and a master node. A slave node is responsible for storing data, while a master node stores metadata. If anyone wants to store or

process the data, than he or she must submit a request to the master node. The master processes that request and sends it to the appropriate node. All customer information can be accessed by the sensors, meaning that the system is very easy to use and it is easy to generate quasistructured data. All of these data must be in an encrypted form. This design security scheme is based on public key infrastructure (PKI) and the RSA [Rivest–Shamir–Adleman] algorithm, ensuring that only authorized users can access a particular patient's data at a certain time. There are two types of data storage in the cloud: master storage and slave storage. The master storage holds metadata, and the slave storage holds electronic patient records (EPRs) and consists of medical and health data. The master storage can be accessed through the data access service (DAS) containing the REST service for the application client. The master storage holds the EPR, which is encrypted with the patient's public key so that only the personal record application on the patient's device can decrypt it with the patient's own private key. The proxy storage holds the EPR as it is requested by the monitoring application. The EPR's metadata in the master storage are encrypted with the public key of those who request and have permission to access it, then it is signed by the patient's private key for integrity and authenticity checks. Once a doctor or medical staff member retrieves the EPR using the monitoring application, he or she verifies the EPR using the patient's public key to prove its authenticity, before decrypting it with his or her own private key [2]. Figure 10.5 illustrates our proposed architecture. All requests will go on the master where the metadata will be located. After that, the master system will send the request to the slave and the slave will respond to the master again. One master can easily manage many slaves.

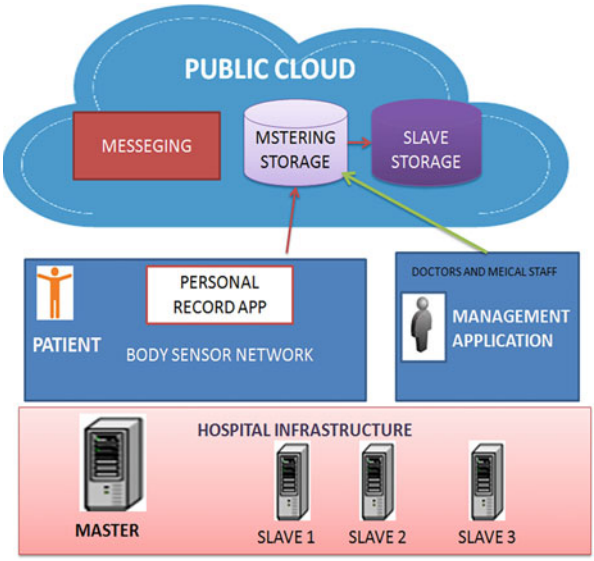


Fig. 10.5 Secured architecture for the healthcare cloud

10.8.1 Results

The messaging between users and the cloud's server occurs on the Secure Sockets Layer (SSL), which ensures the privacy and integrity of message sending and receiving between two parties. But the public server is consider nonliable as it is operated and preserved by the cloud provider company. The cloud provider has no legal right to access information belonging to the user. There is, however, a potential risk that a cloud-based server could be attacked by a malicious program, which could cause unauthorized data access. The security scheme offers fine-grained access management of encrypted data in the cloud. Furthermore, it also ensures the integrity and authenticity of messages transferred through the cloud between the patient and the doctor. Throughout this chapter we use Ω (a, b) → c to denote the operation of running an algorithm Ω with inputs a, b, ... and output c [27].

The key manager generates key pairs and keeps and provides public keys for different users involved in the application system. The access control contains policies that enable personal record applications to validate who can access which patient's data at what level (e.g., pulmonology doctor has read-write access to data on patients with lungs disease, while nurses have only read access). The patient can supervise the access policy on his or her records to take full access control of his or her own data. In a critical situation, that control policy can be overridden other medical staff for a short time. The hospital information system (HIS) is integrated into our organization to provide patients' health records. The key manager, access control, and HIS are hosted by the hospital infrastructure to minimize safety risks. The messaging service on the cloud support sends a notification when access to the EPR is requested or when the latest updated data are available in proxy storage [28].

The EPR has two parts:

1. *Health data*, which are quasistructured data and come from sensors.
2. *Medical data*, which come from the medical staff's personal information, such as their ID, name, etc. When we need to use the record application, a new patient is registered on this system. We can understand what happens in the background during registration, as follows:

   (a) The key manager executes KeyGen() → (Private[P], Public[P]) to generate a key pair for the patient, using the RSA algorithm. Here, P stands for "patient."
   (b) Private [P] is securely stored on the patient's system using the AES algorithm to protect authentication.
   (c) The EPR is loaded from the HIS and encrypted with Encrypt(KEY, Public[P], Tn) → KEYPublict[P], where T represents the data attribute of a vital sign and n is the number of attributes to be encrypted.
   (d) KEYPubP is saved in the master storage through the DAS.
   (e) Finally, the master processes these data, normalizes them, classify them, and sends them to the slave for storage.

For retrieving the data, our proposed architecture will request the credentials by using the visualizing application. The visualizing application supports direct access application (DAA) for an authorized person by which he or she can retrieve the EPR. This DAA is used to decrypt the information, using the public key. If the permission is verified, the following steps will be executed:

1. *Decrypt* (KEYPublic[P], Private[P], Tn) → KEY to decrypt the EPR retrieved from the master storage. Then the master will search for these data in the slave to get the data.
2. *Encrypt* (KEY, Public[R], Sn) → KEYPublic[R] to encrypt with the requester's public key. Here, R stands for the user who is using this application.
3. *Sign* (Private[P], T) → TPrivate[P] to sign a generated hash key denoted by T with the patient's private key to ensure the authenticity of the EPR before sending it together with KEYPublic[R] to the slave storage.

10.9 Conclusion

From the discussion in this chapter, one can easily understand healthcare security issues, healthcare responsibility, and how we can secure our information in the healthcare cloud. With time, we can modify our architecture to make data more secure in the healthcare cloud. Therefore, use of cloud computing in healthcare systems makes health services more affordable, as well as helping nations to achieve health equity. In this chapter, cloud computing and the healthcare cloud have been introduced. Furthermore, cloud computing security issues, particularly in the context of the healthcare cloud, have been presented. This chapter has also proposed and discussed some methods to improve cloud security for healthcare along with our proposed architecture.

References

1. Kim, H., Lee, H., Kim, W., Kim, Y.: A trust evaluation model for QoS guarantee in cloud systems. Int. J. Grid Distrib. Comput. **3**, 125 (2010)
2. Yang, Z., et al.: A collaborative trust model of firewall-through based on cloud computing. 14th International Conference on Computer Supported Cooperative Work in Design, 2010, China
3. Ahmed, M.: Above the trust and security in cloud computing: a notion towards innovation. IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2010, Australia
4. Brodkin, J.: Gartner: Seven cloud-computing security risks, InfoWorld, 2008. https://www.infoworld.com/article/2652198/security/gartner%2D%2Dseven-cloud-computing-security-risks.html. Accessed 15 July 2018
5. Chen, D., Zhao, H.: Data security and privacy protection issues in cloud computing. Int. Conf. on Comput. Sci. Elect. Eng. **1**, 647–651 (2012)
6. Grobauer, B., Walloschek, T., Stocker, E.: Understanding Cloud Computing Vulnerabilities. IEEE Secur. Priv. Mag. **9**(2), 50–57 (2011)

7. Altowaijri, S., Mehmood, R., Williams, J.: A quantitative model of grid systems performance in healthcare organisations. Int. Conf. on Intellig. Syst. Model. Simulat. Liverpool, United Kingdom, pp. 431–436 (2010)
8. Mehmood, R., Faisal, M.A., Altowaijri, S.: Future networked healthcare systems: a review and case study. Handbook of research on redesigning the future of internet architectures, pp. 531–558, (2015)
9. Alazawi, Z., Alani, O., Abdljabar, M.B., Altowaijri, S., Mehmood, R.: A smart disaster management system for future cities. Proceedings of the 2014 ACM international workshop on Wireless and mobile technologies for smart cities - WiMobCity'14, Philadelphia, Pennsylvania, USA, 2014, pp. 1–10
10. Alazawi, Z., Abdljabar, M.B., Altowaijri, S., Vegni, A.M., Mehmood, R.: ICDMS: An Intelligent Cloud Based Disaster Management System for Vehicular Networks. In: Vinel, A., Mehmood, R., Berbineau, M., Garcia, C.R., Huang, C.-M., Chilamkurti, N. (eds.) Communication Technologies for Vehicles, vol. 7266, pp. 40–56. Springer, Berlin, Heidelberg (2012)
11. Alazawi, Z., Altowaijri, S., Mehmood, R., Abdljabar, M.B.: Intelligent disaster management system based on cloud-enabled vehicular networks. In 2011 11th International Conference on ITS Telecommunications, St. Petersburg, Russia, pp. 361–368 (2011)
12. Mehmood, R., Alam, F., Albogami, N.N., Katib, I., Albeshri, A., Altowaijri, S.M.: UTiLearn: A Personalised Ubiquitous Teaching and Learning System for Smart Societies. IEEE Access. **5**, 2615–2635 (2017)
13. Oh, S., et al.: Architecture Design of Healthcare Software-as-a-Service Platform for Cloud-Based Clinical Decision Support Service. Healthc. Inform. Res. **21**(2), 102 (2015)
14. Zhang, R., Liu, L.: Security models and requirements for healthcare application clouds. 2010 IEEE 3rd International Conference on Cloud Computing, Miami, FL, USA, pp. 268–275 (2010)
15. Wan, J., Zou, C., Ullah, S., Lai, C.-F., Zhou, M., Wang, X.: Cloud-enabled wireless body area networks for pervasive healthcare. IEEE Netw. **27**(5), 56–61 (2013)
16. Barton, J., et al.: Impact of cloud computing on healthcare V2.0| Object Management Group. https://www.omg.org/cloud/deliverables/impact-of-cloud-computing-on-healthcare.htm. Accessed 29 June 2018
17. Ahmed, M., Xiang, Y., Ali, S.: Above the trust and security in cloud computing: a notion towards innovation. In 2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, Hong Kong, China, pp. 723–730 (2010)
18. Bildosola, I., Río-Belver, R., Cilleruelo, E., Garechana, G.: Design and Implementation of a Cloud Computing Adoption Decision Tool: Generating a Cloud Road. PLoS One. **10**(7), e0134563 (2015)
19. Balasubramaniam, S., Kavitha, V.: Geometric Data Perturbation-Based Personal Health Record Transactions in Cloud Computing. Sci. World J. **2015**, 1–9 (2015)
20. Elizabeth, S.: HealthITSecurity, data security considerations in healthcare interoperability. HealthITSecurity. https://healthitsecurity.com/features/data-security-considerations-in-healthcare-interoperability. Accessed 24 June 2018
21. Saurabh: Security issues in cloud Computing. http://serl.iiit.ac.in/cs6600/saurabh.ppt. (2009). Accessed 17 June 2018
22. Sherry, D.: Cloud computing: security risks and compliance implications. http://media.techtarget.com/searchFinancialSecurity/downloads/FISD09_Breakout_Session5_ CloudComputing_Sherry.pdf, Brown University (2009). Accessed 14 June 2018
23. Ryoo, J., Rizvi, S., Aiken, W., Kissell, J.: Cloud Security Auditing: Challenges and Emerging Approaches. IEEE Secur. Priv. **12**(6), 68–74 (2014)
24. Kwon, J., Johnson, M.E.: Protecting patient data-the economic perspective of healthcare security. IEEE Secur. Priv. **13**(5), 90–95 (2015)
25. Schoo, P., et al.: Challenges for cloud networking security. In Mobile Networks and Management, 2011, pp. 298–313
26. Yang, Z., Qiao, L., Liu, C., Yang, C., Wan, G.: A collaborative trust model of firewall-through based on Cloud Computing. The 2010 14th International Conference on Computer Supported Cooperative Work in Design, Shanghai, China, pp. 329–334 (2010)

27. Goyal, S.: 5 reasons why you should choose multi-tenant architecture for your SaaS application. Insights—Web and Mobile Development Services and Solutions, https://www.netsolutions.com/insights/5-reasons-why-you-should-choose-multi-tenant-architecture-for-your-saas-application/. Accessed 11 July 2018
28. Shaikh, R., Sasikumar, M.: Security Issues in Cloud Computing: A survey. Int. J. Comput. Appl. **44**(19), 4–10 (2012)
29. Rathi, G., Abinaya, M., Deepika, M., Kavyasri, T.: Healthcare data security in cloud computing. IJIRCCE, 3(3), (2015). ISSN(Online): 2320-9801 ISSN (Print): 2320-9798
30. Almond, C.: A practical guide to cloud computing security, (2009). http://www.avanade.com/Documents/Research%20anad%20Insights/practicalguidetocloudcomputingsecurity574834.pdf. Accessed 11 July 2018
31. Chondamrongkul, N., Chondamrongkul, P.: Secure mobile cloud architecture for healthcare application. Int. J. Fut. Comput. Commun. **6**(3), 77–86 (2017)

Chapter 11
The Role of Big Data and Twitter Data Analytics in Healthcare Supply Chain Management

Shoayee Alotaibi, Rashid Mehmood, and Iyad Katib

11.1 Introduction

The healthcare sector is considered one of the main economic pillars worldwide, to which significant proportions of countries' budgets are allocated. It is estimated that healthcare spending in the world's major regions will increase from 2.4 to 7.5% of GDP between 2015 and 2020 [1]. Despite this massive expenditure, healthcare organizations are required to deliver high-quality medical services at lower costs to their patients. However, spending hundreds of millions does not alone guarantee high-quality services. Hence, most of healthcare organizations nowadays are faced with incremental challenges, including limited budget, daily increases in patient numbers and increasing costs of medical equipment and pharmaceuticals [2]. As much as 45 percent of a hospital's typical total operating expense is committed to its supply chain, including suppliers, drugs, and consumables.

The healthcare supply chain is an essential area that should be considered and improved. It would be incorrect to understand it as only relating to purchasing and managing contracts, as it is a very complex concept, and could free up huge revenues within healthcare sectors once managed properly [3]. Consequently, healthcare organizations will likely increasingly need to employ recent technological developments to deliver efficient services at lower costs and high quality. Moreover, such improvements are required to reduce the waste and loss that threaten sustainability.

S. Alotaibi (✉) · I. Katib
Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz Univeristy, Jeddah, Saudi Arabia
e-mail: salotaibi0372@stu.kau.edu.sa; iakatib@kau.edu.sa

R. Mehmood
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: RMehmood@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*, EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_11

In the current era of increasingly advanced technologies in medical devices and medical equipment, the size of data generated by their use is growing exponentially. The immense growth in the volume of electronic medical records (EMRs) stored by healthcare organizations is also significant and undeniable. Exploring the possibility of investing this big data in improving services has become attractive to researchers and practitioners. A lot of fruitful business applications and network search engines have been developed using Business Intelligence (BI) for extracting knowledge from big data [4]. Some researchers have been investigating how to transfer and where to store this amount of data, while others have been focusing on big data utilization. Big data utilization involves analysing it to seek a solution for existing issues, exploring trends, and supporting decision-making.

A plethora of literature has been produced that explores to what extent big data can be beneficial in the healthcare industry. Malik and his colleagues [5] noted that big data analytics seems to have been frequently used for the diagnosis, prognosis or planning of treatment, for example, disease management for oncology to anticipate heart attacks and identify and classify at-risk people. However, a very limited work has been done in applying big data to healthcare supply chains. Existing published survey papers focus on reviewing the significant applications of big data to supply chains in manufacturing generally.

In this paper, our aim is to review research on the use of big data in the healthcare supply chains. We will investigate the opportunities, challenges and future directions of big data in this field. This is an extended version of our earlier work [6].

The chapter is organized as follows: Section 11.2 gives brief definitions for the basic concepts that are mentioned in this paper. The next section highlights the published works in big data analytics and Twitter data analytics. Big data analytics in supply chain and healthcare opportunities and challenges are discussed in Sect. 11.4. Section 11.5 concludes the work and suggests possible future directions.

11.2 Background

This section introduces the work by defining the basic terminologies that are mentioned in this paper. Supply chain, big data and big data analytics are illustrated based on the reviewed references.

11.2.1 Supply Chain

Malik et al. define the supply chain process as “having the right item in the right quantity at the right time at the right place for the right price in the right condition to the right customer” [7]. In the meantime, supply chain managers can legitimately claim to have played a major role in spreading the information technology revolution. E-SCM (e-supply chain management) was a great transformation as supply

chain activities were integrated with the Internet [8]. Smarter supply chains [9] and smart factories [10] are further examples of intelligent systems developments. Sustainability (triple bottom line, TBL) has become a crucial consideration in business, government and academia. Therefore, the concept and practice of green or sustainable supply chains have become a vital part of industrial and government operations, see e.g [11, 12].

11.2.2 Supply Chain Activities in Healthcare

Based on the above section, supply chain management in healthcare is not limited only for pharmaceutical products or physical stuff, it is also involving the life time of the service that is delivered to the patients. In this context, supply chain management in healthcare is a very complex and interrelated process. It's mainly targeted to deliver adequate and efficient heath care to the patient. In order to that, several activities are performed including medical equipment and pharmaceutical products requests from the manufacturers and service delivery procedure inside the health organization. Hence, a number of independent stakeholders are involved in these activities such as manufacturers, insurance companies, hospitals staff and several regulatory agencies (Fig. 11.1).

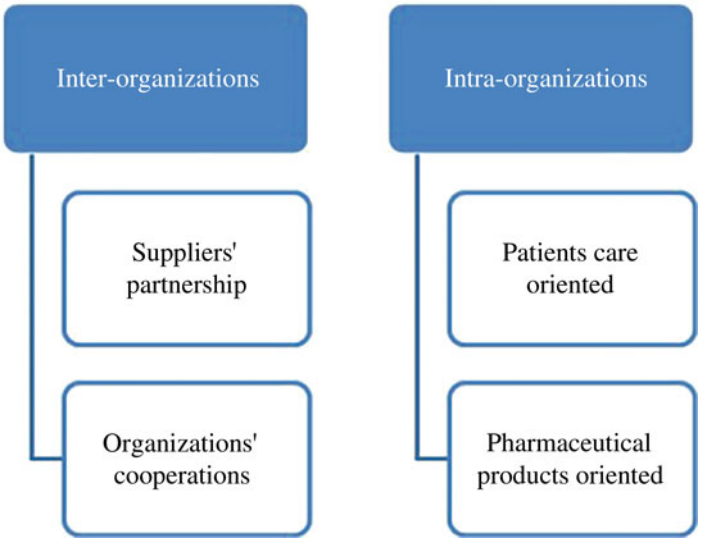


Fig. 11.1 Supply chain management activities in healthcare

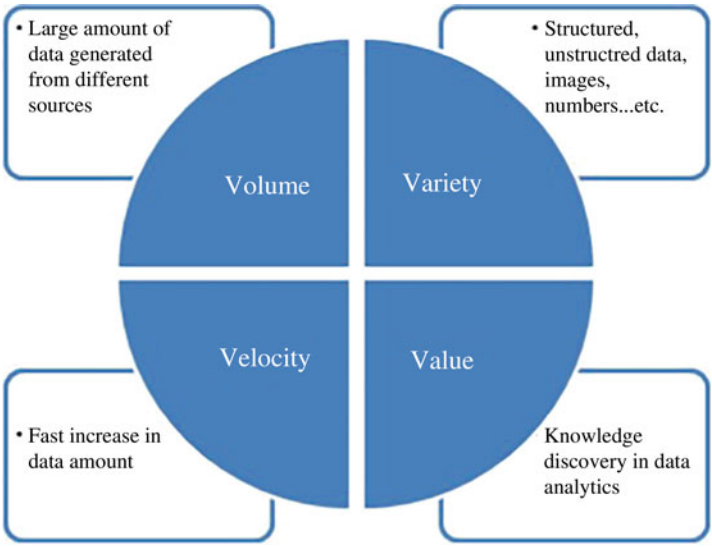


Fig. 11.2 Big data 4Vs

11.2.3 Big Data

According to [4], big data refers to "the datasets that could not be perceived, acquired, managed, and processed by traditional IT and software/hardware tools within a tolerable time". However, researchers and scientists have defined the term "big data" according to several different aspects. Apache Hadoop in 2010 defined big data as "datasets which could not be captured, managed, and processed by general computers within an acceptable scope".

In 2011, an IDC report characterized big data as "large information innovations depict another era of advancements and structures, intended to financially extricate an incentive from substantial volumes of a wide assortment of information, by empowering the high-speed catch, disclosure, as well as examination". Big data technologies have also been defined as "the emerging technologies that are designed to extract value from data having four Vs characteristics; volume, variety, velocity and veracity" [13]. Accordingly, the key attributes of big data can be outlined as the "four Vs", i.e. Volume (extraordinary volume), Variety (different modalities), Velocity (quick era), and Value, as shown in Fig. 11.2.

11.2.4 Big Data Analytics

Big data analytics has become a key buzzword these days. It is not just a buzzword but is making a fundamental impact on all spheres of our life, transport [14],

planning and operations [15, 16], smart cities [17], teaching and learning [18], to name but a few. According to Feki and Wamba [19] and Hogarth and Soyer [20], the term "analytic" can be defined as transforming big data into meaningful intelligent information. This transformation of big data is usually done using two main steps: data management, then data analytics using specific techniques [21]. Data management implies "processes and supporting technologies to acquire and store data and to prepare and retrieve it for analysis" while analytics means "techniques used to analyse and acquire intelligence from big data" [19].

11.2.5 Twitter Data

Twitter is one of the most popular social media networking that attracted many users around the world. Recently, the number of Twitter users is relatively increasing with the growth of smart phones technology. Accordingly, Twitter attracted many researchers in different industries to investigate the efficiency of such pool of open data in enhancing services or reducing service expenses. That's due to availability of API tools that enable researchers to collect tweets using specific keywords.

The most common steps in Twitter data analytics research can be summarized as follows:

1. **Tweet Collection** Tweets can be collected using one of the available API tools. The collection can be for historical tweets or live streaming for the tweets.
2. **Pre-processing Tweets** This step is very important to remove the irrelevant and meaningless tweets. In case of sentiment analysis, some additional steps such as tokenization are also added.
3. **Data Analysis** This is the core step where the resulting tweets can be analysed. Different type of analysis can be applied such as descriptive analysis (DA), content analysis (CA) and network analysis (NA).
4. **Reporting Results** This is the last step where the results of analysis are reported. Therefore, the researchers can get answers for their research questions based on the analysis results (Fig. 11.3).

Fig. 11.3 Twitter data analysis steps

11.3 Big Data and Twitter Data in Healthcare or Supply Chain Management

In this section, some examples of how big data has been used in healthcare or supply chain, individually, are provided. Moreover, some contributions on using Twitter data in healthcare and supply chain practices are highlighted.

11.3.1 Big Data in Supply Chain Management

Big data has been widely used in supply chain management in many industries. According to Waller & Fawcett [22], despite the operational influence of big data in supply chains, traditional approaches and standard activities are affected, too. They identified the potential opportunities that big data could offer in enhancing supply chain processes.

Carriers, manufacturers and retailers, the main users of logistics, are also the main beneficiary of big data. They could obtain actionable information about many of their daily activities, such as inventory, transport and human resources management. DHL and UPS are two leading companies who are pioneer investors in big data initiatives to enhance their services and increase their profits [23].

Further attempts on investigating the big data practices in supply chain management could be found in [14, 24–29].

11.3.2 Big Data in Healthcare

In healthcare industry, there are many sources for the big data as shown in Fig. 11.2. The use of big data is not limited to industrial fields. It is playing a key role in enhancing critical service sectors such as healthcare. Healthcare systems and applications have long been considered computationally intensive [30]. However, the focus on data—i.e. big data—has only began in the last few years. It As noted in [13], "the cost of healthcare, according to World Health Organization is mostly due to system and operational inefficiencies, and missed disease-prevention opportunities. Big data analytics can minimize these efficiencies and improve the clinical processes resulting in better, preventive, personalized healthcare; estimated to save billions in the healthcare sector alone with virtually unquantifiable impact".

Collaborations between big data platform providers and scientific research centres have generated remarkable and noticeable successes. In Australia, two innovative applications for big data have been developed by Srinivasan and Arunasalam [31]. They have utilized the massive data extracted from hospital discharge reports and insurance claims to detect fraud, abuse, waste and errors in insurance claims.

Fig. 11.4 Examples of big data sources in healthcare

Similarly, in 2014, Raghupathi1and Raghupathi [32] reported that in healthcare more than $300 billion could be saved annually through big data analytics utilization, as estimated by McKinsey. Big data utilization could be applied in two vital areas: Clinical Operations and Research & Development [33]. A practical example of using big data analytics has been undertaken by developers [34] in US healthcare sector. They built predictive systems based on big data that could help in early identification of six critical cases: high cost patients, readmissions, triage, decompensating (once a patient's situations get worse), adverse events, and treatment optimization for diseases affecting multiple organ systems.

Several works exist that use big data to improve healthcare ICT systems efficiencies. For example, the use of cloudlets and big data to improve mobile healthcare systems response and experience is proposed in [35, 36]. A capacity sharing model for healthcare using big data is proposed in [37]. The use of big data to improve the performance of networked (integrated) healthcare systems is proposed in [13]. A smart pain management system using big data computing is proposed in [38]. DNA profiling is an emerging application of big data [39] (Fig. 11.4).

11.3.3 Twitter Data Analytics in Supply Chain Management

Social media platforms such as Twitter are considered as main sources of big data where a large number of users share their daily life. A large and growing body of literature has investigated the usefulness of Twitter in public health. The majority of the contributions in this field is either surveillance or prediction tools for certain disease or discovering disease pattern in specific community.

To the best of our knowledge, only one published work we found which focused on utilizing Twitter data in supply chain management industry. The only published contribution has been conducted by Chae in [40]. In his research, he proposed a framework to investigate the potential role of using Twitter for supply chain practices. Three different analysis methodologies have been applied on the collected tweets from #supplychain hashtag. The findings show the possibility of harvesting Twitter data in demand shaping, cost reducing or improving the service quality in supply chain activities. This research can encourage researchers to investigate the possible ways to take full advantages of Twitter data in supply chain in particular.

In 2013, Alex and his colleagues [41] enhance the tracking of flue infection by providing deeper content analysis for the tweets. They distinguished between the awareness and infection related tweets to provide more accurate counts for the flue infection cases. Similarly, Armaki et al. in [42] enhanced the performance of a surveillance tool by developing a support vector machine (SVM) classifier that is able to catch the flue infection through tweets. Moreover, the Social Network Enabled Flu Trends (SNEFT), a constant data collection framework have been affirmed to indicate the influenza dispersal through recording flu relevant tweets [43].

Much attention has been paid to influenza in particular as it was reported as one of death causes in United States recently. In this context, a global real-time tool that is able to anticipate the flue infection in specific area has been built based on the extracted information from Twitter as well [44]. Another opportunity is using the number of retrieving medical related articles as indicator to seasonal infections [45].

11.4 Big Data in Healthcare Supply Chains

In this section, we demonstrate the possible opportunities of using big data as a solution in healthcare supply chains. The opportunities are summarized based on the previous works that have been published. Unfortunately, very limited work has been found. However, the application of big data in this regard is unlimited and further investigations are required. In the last subsection, the challenges that might be considered are listed.

11.4.1 Opportunities

Nowadays, big data has in many ways become a solution looking for a problem to solve. Rozados and Tjahjono saw that “Major business players who embrace big data as a new paradigm are seemingly offered endless promises of business transformation and operational efficiency improvements” [29]. This has attracted researchers and practitioners in many industries to explore the possibilities of using big data. Abundant research has been done in both healthcare and in supply chain

management generally. The healthcare industry is considered as an essential and critical sector within services, but there is a lack of information about the current state of research into healthcare operations management (OM) and supply chain management (SCM) [46]. At the time of writing this paper, only three peer reviewed papers have been found in this area, and we can summarize the opportunities of using big data in healthcare supply chains as follows.

Strategic Planning Big data represented by Twitter data can be harvested in several ways. Open access source of pool data could help in shaping supply chain management activities. At early stages, public health screening for specific nations within certain geographical area be gained by analysing Twitter data appropriately. Identifying list of health concerns leads to determine the type and location of delivered health services. Moreover, real-time data analytics for Twitter data can support some statistics of disease infections.

Disaster and Risk Management In disaster management activities, monitoring public discussions on Twitter hashtags during some occasions would support statistics about health cases. That's would help the organizations and health service providers to ensure their readiness to urgent cases containment. They could estimate their needs and eliminate the over/understocking.

Demand Forecasting At management level in many industries, demand forecasting is widely used in order to decision-making reinforcement and to promote other management tasks. In China, historical recorded data from transaction datasets has been successfully used to build a predictive model based on data mining algorithms [47]. This model is supposed to work as a prediction tool to estimate future needs within the healthcare supply chain process in China. They used real datasets from 2014 to build the prediction tool, to predict the next year's needs. Since the nature of the collected data set is heterogeneous, and in order to empower the prediction tool, they combined a classification decision tree and regression algorithm in CRT modelling. The efficiency of their model was proven and gained better results than other traditional statistical approaches.

Improving Safety and Quality Assurance in the Pharmaceutical Supply Chain In the pharmaceutical industry, counterfeiting and illegal export and import of medicines is a major issue. Moreover, transferring medicines and medical equipment in inappropriate environmental conditions, such as at high temperature and humidity, can affect quality. Thus, the challenge is to guarantee the delivery of shipped medicines safely. Further, medical care providers (hospitals, clinics, etc.) need to verify that they have obtained the right medicine from the right source. In Germany XQ in [2] made use of the data stored by their RFID-based system about tracked and traced shipments, such as ID, location, temperature, and humidity. Tracking and tracing are widely known terms in the supply chain management context, which may offer opportunities to ensure quality of medicines and prevent counterfeiting.

Indoor Monitoring For healthcare organizations, the benefits of track and trace systems are not limited to ensuring medicines' quality. Data generated from these systems can also offer an opportunity to improve the safety of special needs patients and new-born babies. A healthcare unit's administrators can retrieve real-time locations and other necessary information, such as vital signs for Alzheimer's patients, at any moment, to ensure that they are safe. Intelligent applications can offer monitoring without restricting patients' movements. Also, new-born babies can be saved from kidnapping and theft. A real application for this opportunity was delivered by Sultanow and Chircu [2] when they launched the track and trace system and reported its significant benefits.

11.4.2 Challenges

While big data could offer a wide range of opportunities, it has characteristics that could be considered as important challenges, both generally as well as in the case of healthcare, specifically. The criticality of the healthcare industry and its standards of confidentiality might create difficulties too. The key challenges of applying big data in the healthcare supply chain can be summarized as follows.

Data Related Issues Due to big data's characteristics, such as data volume, variety, and heterogeneity, some issues may arise. According to Tan et al., the variations of data require finding special techniques for handling and storing, as claimed by Burghin et al. [48]. Moreover, the traditional data mining techniques may not be longer sufficient for such kinds of data [49]. Alongside (and sometimes as a result of) the variety and volume, incompleteness, incorrectness, and uselessness are also commonly reported difficulties.

Healthcare Related Issues The main resources of big data in the healthcare industry are electronic medical records (EMRs) [32]. Practitioners use EMRs to record patient's medication histories every time the patient visits the clinic. According to [6], data ownership, governance and standardization are the main challenges that should be considered in this area.

Knowledge Related Issues Deep knowledge is needed in order to understand the variety of data forms and analyse the relationship between different kinds of data [31]. Moreover, the topic is complexly multidisciplinary, since sufficient knowledge of big data analytics techniques, healthcare data and supply chain processes are required, too.

Data Analytics Tools Related Issues Despite the ease of access to big data, the constraints associated with the available analytics tools could limit the big data investments. Constraints on data collections, expense of storage and the inaccuracy of analysis tools are some of expected issues that analysts might face. More development for the existing data analytics tools is necessary in order to take the full advantages of big data.

11.5 Conclusion and Future Research Directions

In conclusion, healthcare supply chains are an essential area that should be considered and improved. Healthcare organizations will likely need to employ recent developments in technology to deliver efficient services at reasonable cost and high quality. Improved data analysis is also required to reduce the waste and loss that threaten sustainability. Big data analytics is a powerful tool that is usually concerned with large-scale data and high-performance computing environments; it has emerged as a revolution that is able to contribute in different ways to many field, such as through data analysis, knowledge extraction and advanced decision-making. We recommend some future directions for the use of big data in healthcare supply chains in the following.

1. Data driven inventory can enhance prediction tools through several optimization methods. This includes studying how to get benefits from "data patterns" that are extracted at the analysis step, and how to use them to support decision-making.
2. Further reviews of how big data is used in manufacturing, unrelated to patients, is another possible direction, informing how we might use patient-centric data in estimating hospitals' needs or logistic operations such as scheduling, staff scheduling, resources allocation, and hospital design layout.
3. Using social media in addition to EMRs can assist in determining the best locations for future clinics and services.

An important step to enable optimized supply chains in healthcare sector would be the networking and integration of healthcare and other smart world systems [35]. Such integration would give rise to a plethora of useful data where the systems integration would allow automatic collection, storage, and analyses of data. Moreover, the integration would also enable optimized decisions to be taken and enforced automatically leading to optimized supply chains in the healthcare sector.

Acknowledgments The work carried out in this paper is supported by the HPC Center at the King Abdulaziz University.

References

1. Deloitte: 2017 Global Health Care Sector Outlook. 2015 (2017)
2. Sultanow, E., Chircu, A.M.: Improving healthcare with data-driven track-and-trace systems. 65–82
3. Kwon, I.W.G., Kim, S.H., Martin, D.G.: Healthcare supply chain management; strategic areas for quality and financial improvement. Technol. Forecast. Soc. Change. **113**, 422–428 (2016)
4. Chen, M., Mao, S., Liu, Y.: Big data: a survey. Mob. Networks Appl. **19**, 171–209 (2014)
5. Malik, M.M., Abdallah, S., Ala'raj, M.: Data mining and predictive analytics applications for the delivery of healthcare services: a systematic literature review. Ann. Oper. Res. 1–26 (2016)
6. Alotaibi, S., Mehmood, R.: Big data enabled healthcare supply chain management: opportunities and challenges. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart

Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 207–215. Springer, Cham (2018)
7. Naoui, F.: Customer service in supply chain management: a case study. J. Enterp. Inf. Manag. **27**, 786–801 (2014)
8. Akyuz, G.A., Rehan, M.: Requirements for forming an "e-supply chain". Int. J. Prod. Res. **47**, 3265–3287 (2009)
9. Butner, K.: The smarter supply chain of the future. Strateg. Leadersh. **38**, 22–31 (2010)
10. Hessman, T.: The Dawn of the Smart Factory. IndustryWeek. 14–19 (2013)
11. Ahmad, N., Mehmood, R.: Enterprise systems: are we ready for future sustainable cities. Supply Chain Manag. An Int. J. **20**, 264–283 (2015)
12. Ahmad, N., Mehmood, R.: Enterprise systems and performance of future city logistics. 27, 500–513 (2016). doi:https://doi.org/10.1080/09537287.2016.1147098
13. Mehmood, R., Faisal, M.A., Altowaijri, S.: Future Networked Healthcare Systems: A Review and Case Study. In: I. Management Association (ed.) Big Data: Concepts, Methodologies, Tools, and Applications. pp. 2429–2457. IGI Global (2016)
14. Mehmood, R., Meriton, R., Graham, G., Hennelly, P., Kumar, M.: Exploring the influence of big data on city transport operations: a Markovian approach. Int. J. Oper. Prod. Manag. **37**, 75–104 (2017)
15. Suma, S., Mehmood, R., Albugami, N., Katib, I., Albeshri, A.: Enabling Next Generation Logistics and Planning for Smarter Societies. Procedia - Procedia Comput. Sci. 1–6 (2017)
16. Enabling Smarter Societies through Mobile Big Data Fogs and Clouds, http://www.sciencedirect.com/science/article/pii/S1877050917311213
17. Alam, F., Mehmood, R., Katib, I., Albogami, N.N., Albeshri, A.: Data fusion and IoT for smart ubiquitous environments: a survey. IEEE Access. **5**, 9533–9554 (2017)
18. Mehmood, R., Alam, F., Albogami, N.N., Katib, I., Albeshri, A., Altowaijri, S.M.: UTiLearn: a personalised ubiquitous teaching and learning system for smart societies. IEEE Access. **5**, 2615–2635 (2017)
19. Feki, M., Wamba, S.F.: Big Data Analytics-enabled Supply Chain Transformation: A Literature Review. 49th Hawaii Int. Conf. Syst. Sci. 1123–1132 (2016)
20. Hogarth, R.M., Soyer, E.: Using simulated experience to make sense of big data. MIT Sloan Manag. Rev. **56**, 49–54 (2015)
21. Gandomi, A., Haider, M.: Beyond the hype: big data concepts, methods, and analytics. Int. J. Inf. Manag. **35**, 137–144 (2015)
22. Waller, M.A., Fawcett, S.E.: Data science, predictive analytics, and big data: a revolution that will transform supply chain design and management. J. Busienss Logist. **34**, 77–84 (2013)
23. Zhong, R.Y., Newman, S.T., Huang, G.Q., Lan, S.: Big data for supply chain management in the service and manufacturing sectors: challenges, opportunities, and future perspectives. Comput. Ind. Eng. **101**, 572–591 (2016)
24. Samuels, K.: Practitioners understanding of big data and its applications in supply chain management. Electron. Libr. **35**, 616–617 (2017)
25. Lamba, K., Singh, S.P.: Big data in operations and supply chain management: current trends and future perspectives. Prod. Plan. Control. **28**, 877–890 (2017)
26. Brinch, M., Stentoft, J.: Big data and its applications in supply chain management: findings from a Delphi Study. 1351–1360 (2017)
27. Schoenherr, T., Speier-Pero, C.: Data science, predictive analytics, and big data in supply chain management: current state and future potential. J. Bus. Logist. **36**, 120–132 (2015)
28. Benabdellah, A.C., Benghabrit, A., Bouhaddou, I., Zemmouri, E.M.: Big Data for Supply Chain Management: Opportunities and Challenges. 7, 20–26 (2016)
29. Varela, I.R., Tjahjono, B.: Big data analytics in supply chain management: trends and related research. 6th Int. Conf. Oper. Supply Chain Manag. 1, 2013–2014 (2014)
30. Altowaijri, S., Mehmood, R., Williams, J.: A quantitative model of grid systems performance in healthcare organisations. In: ISMS 2010 - UKSim/AMSS 1st International Conference on Intelligent Systems, Modelling and Simulation. pp. 431–436 (2010)

31. Srinivasan, U., Arunasalam, B.: Leveraging big data analytics to reduce healthcare costs. IT Prof. **15**, 21–28 (2013)
32. Raghupathi, W., Raghupathi, V.: Big data analytics in healthcare: promise and potential. Heal. Inf. Sci. Syst. **2**, 3 (2014)
33. Feldman, B., Martin, E.M., Skotnes, T.: Big Data in healthcare - hype and hope, http://www.riss.kr/link?id=A99883549, (2012)
34. Bates, D.W., Saria, S., Ohno-Machado, L., Shah, A., Escobar, G.: Big data in health care: using analytics to identify and manage high-risk and high-cost patients. Health Aff. **33**, 1123–1131 (2014)
35. Tawalbeh, L.A., Mehmood, R., Benkhlifa, E., Song, H.: Mobile cloud computing model and big data analysis for healthcare applications. IEEE Access. **4**, 6171–6180 (2016)
36. Tawalbeh, L.A., Bakhader, W., Mehmood, R., Song, H.: Cloudlet-Based Mobile Cloud Computing for Healthcare Applications. In: 2016 IEEE Global Communications Conference (GLOBECOM). pp. 1–6. IEEE (2016)
37. Mehmood, R., Graham, G.: Big data logistics: a health-care transport capacity sharing model. Procedia Comput. Sci. **64**, 1107–1114 (2015)
38. Al Shehri, W., Mehmood, R., Alayyaf, H.: A smart pain management system using big data computing. In: Mehmood R., Bhaduri B., Katib I., Chlamtac I. (eds) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. pp. 232–246 (2018)
39. Alamoudi, E., Mehmood, R., Albeshri, A., Gojobori, T.: DNA profiling methods and tools: A review. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST. pp. 216–231. Springer, Cham (2018)
40. Chae, B.: Insights from hashtag #supplychain and twitter analytics: considering twitter and twitter data for supply chain practice and research. Int. J. Prod. Econ. **165**, 247–259 (2015)
41. Lamb, A., Paul, M.J., Dredze, M.: Separating fact from fear: Tracking flu infections on Twitter. Proc. NAACL-HLT 2013. 789–795 (2013)
42. Aramaki, E.: Twitter catches the flu: detecting influenza epidemics using twitter the University of Tokyo the University of Tokyo National Institute of. Comput. Linguist. **2011**, 1568–1576 (2011)
43. Achrekar, H., Gandhe, A., Lazarus, R., Yu, S., Liu, B.: Twitter improves seasonal influenza prediction. Proceding Heal. Informatics. 61–70 (2012)
44. Broniatowski, D.A., Paul, M.J., Dredze, M.: National and local influenza surveillance through twitter: an analysis of the 2012-2013 influenza epidemic. PLoS One. **8**, e83672 (2013)
45. Parker, J., Wei, Y., Yates, A., Frieder, O., Goharian, N.: A framework for detecting public health trends with Twitter. Proc. 2013 IEEE/ACM Int. Conf. Adv. Soc. Networks Anal. Min. - ASONAM '13. 556–563 (2013)
46. Dobrzykowski, D., Saboori Deilami, V., Hong, P., Kim, S.C.: A structured analysis of operations and supply chain management research in healthcare (1982-2011). Int. J. Prod. Econ. **147**, 514–530 (2014)
47. Xu, S., Tan, K.H.: Data-driven inventory management in the healthcare supply chain. (2016)
48. Bughin, J., Chui, M., Manyika, J.: Clouds, big data, and smart assets: Ten tech-enabled business trends to watch. McKinsey Q. 75–86 (2010)
49. Tan, K.H., Zhan, Y.Z., Ji, G., Ye, F., Chang, C.: Harvesting big data to enhance supply chain innovation capabilities: an analytic infrastructure based on deduction graph. Int. J. Prod. Econ. **165**, 223–233 (2015)

Part III
Miscellaneous Applications

Chapter 12
A Mobile Cloud Framework for Context-Aware and Portable Recommender System for Smart Markets

Aftab Khan, Aakash Ahmad, Anis Ur Rahman, and Adel Alkhalil

12.1 Introduction

Smart city systems are an emerging trend that utilize information and communication technologies (ICTs) to offer improved urban services to individuals and collectively refining the lifestyle of societies [11]. In recent years, research and practices have intended to transform the traditional cities and societies into technology and knowledge-driven twenty-first century metropolis [4, 10]. In the context of smart city systems, mobile computing has emerged as a pervasive technology that has empowered its users—with mobility and context-awareness—to accomplish a range of tasks including portable computation as well as location-aware communication [31, 33]. Mobile computing provides the users with mobility-driven and context-aware interfaces to select and utilize the available services including but not limited to smart health, transportation, business, and socialization offered by smart city systems [37].

A. Khan
School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad, Pakistan
e-mail: 13mscsmaftab@seecs.edu.pk

A. Ahmad (✉) · A. Alkhalil
College of Computer Science and Engineering, University of Ha'il, Ha'il, Saudi Arabia
e-mail: a.abbasi@uoh.edu.sa; a.alkalel@uoh.edu.sa

A. U. Rahman
School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad, Pakistan

Department of Information System, Faculty of Computer Science and Information Technology, University of Malaya, Malaysia
e-mail: anis.rahman@seecs.edu.pk

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_12

Mobility or portability is regarded as one of the central features of mobile computing that also provides the foundation for context-aware computing [37]. However, mobility also enforces resource constraints such as limited hardware that affects computation, storage, and energy-related tasks on mobile devices. There is a need for solutions that maintain the balance between mobility and resource availability in the context of mobile computing and smart city systems [2]. In contrast to mobile computing, cloud computing model exploits the "pay-per-use" services model to provide virtually unlimited processing and storage resources [24]. Cloud computing offers the entities or organizations to off-load or deploy their (on-premise) software systems, computations, or storage resources to remote servers by means of cloud-based services [17]. For example, the research in [22] highlights an approach known as cyber-foraging that off loads the computation/storage intensive tasks from a mobile device to (cloud-based) servers in order to enhance computation and energy efficiency of mobile devices. In the context of resource-constrained mobile computing, resource-sufficient cloud computing can be viewed as an opportunistic model that allows mobile devices to compensate their resource poverty by offloading mobile data and computation to cloud servers [34]. Therefore, the unification of mobile and cloud computing can benefit from the mobility and context-awareness of mobile computing, and the computation/storage services of cloud computing to provide systems that are portable and resource sufficient [21].

Research Context In the past, electronic commerce (e-commerce) systems have proven to be useful by digitally offering products and services to international marketplaces. In the current era, business systems/entities heavily rely on reaching their potential customers by recommending them highly customized products and offers. Now with mobile commerce (m-commerce), the use of context information such as age, gender, preferences, and location to offer customer recommendations has gained a significant attention [39]. For example, based on the users' contextual information, any software that provides recommendations such as socialization activities, product and service offerings, and dining options enables human decision support and gives rise to smart systems. Considering a wide-spread adoption of the mobile and cloud computing, there is still a lack of solutions that facilitate its users with a recommendation of their preferences primarily based on their localized context. One of the main challenges for managing and exploiting context-aware recommendations is to identify the contextual factors (such as location and preferences) that influence decisions and actions of people in smart city context [5].

Recommender systems represent a class of software systems that generate meaningful recommendations of interests for their users and empower the users with decision support [28, 32]. Context-aware recommender systems provide dynamic adaptive recommendations to users based on contextual information such as the location, gender, and other preferences of the user [36]. In a smart city context, *smart markets* refer to electronic (virtualized) marketplaces that exploit ICT technologies and infrastructure to enable or enhance digitized commerce. Recommendations and matchmaking between potential customers and business entities are enabled by the relevancy of contextual information that gives rise to the concept of smart markets. Such markets offer personalized offers, products, services, and delivery by business

entities to potential customers while minimizing the irrelevant mass publicity. For instance, by calculating user's location, age, and other relevant information, a mobile recommender system acts as a context-sensing and portable computer that can notify the user on the go about his/her events or places of interests. Recently, much research and development is being carried out to support context-aware recommender systems. However, existing mobile recommender systems fall short of context-aware recommendations in smart city systems to support the activities of smart markets [27, 32, 43].

Solution Overview We overview the proposed recommender system based on the illustrations in Fig. 12.1 that also highlights the activities of smart markets. The proposed framework has two layers, namely: (1) *front-end mobile computing layer* and (2) *back-end cloud computing layer*. By acting as the front-end layer of the framework, a mobile device plays two distinct roles that include (1) sensing the user's context (i.e., location, age, and preferences) and (2) providing an interface to display context-aware recommendations. Cloud computing acts as the back-end layer to compensate for limited resources of a mobile device by providing storage

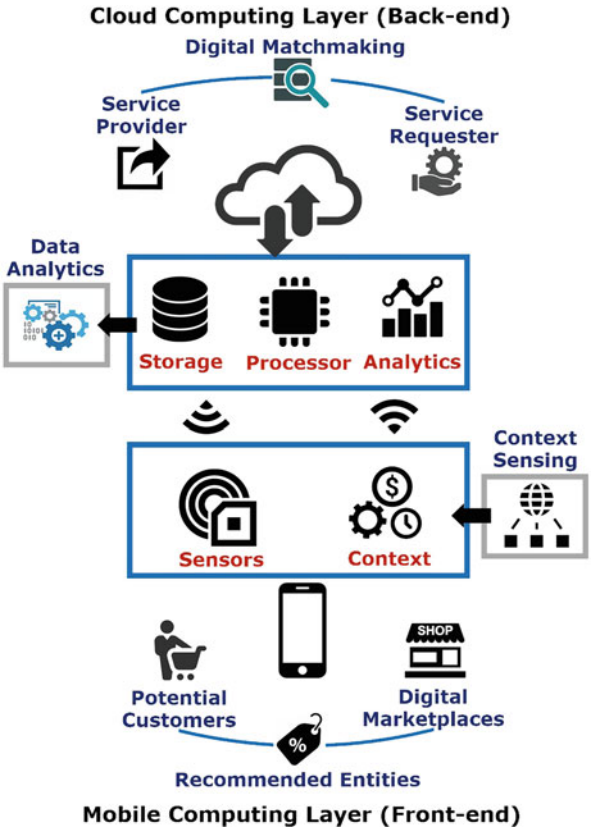


Fig. 12.1 An overview of the proposed mobile cloud recommender system

and computation resources to produce recommendations that are communicated to the mobile device. The proposed solution[1] allows markets, businesses, and transaction-driven entities to communicate with their potential customers in a smart way, i.e., to provide recommendations to the customers based on their localized context of location and preferences. The primary challenge to generate context-aware recommendations lies with the identification of contextual elements such as user's location and preferences from different sources. Another research challenge for recommendation systems is to yield recommendations in real-time fashion for a given user from large and diverse dataset(s) of persons' past preferences. In doing so, there is a need to efficiently utilize the resource-constrained mobile devices that can execute the computation and energy-intensive tasks efficiently.

Proposed Contributions A recent survey on the existing mobile recommender systems highlights that current solutions fall short of context-aware recommendations in a smart market domain [32]. The proposed solution introduces a system architecture, algorithms, enabling technologies, and a prototype for mobile-cloud-based context-aware recommender system. We outline the primary contributions of the proposed solution as:

- Unification of the mobile and cloud computing technologies to provide a framework that supports users' decision support in smart city systems. The framework empowers its users with mobility and context-awareness while processing complex recommendations accurately and efficiently.
- Algorithms and prototype that support automation and user-based customization to provide portable and context-aware matchmaking between potential customers and business entities in smart markets.
- Exploiting mobile cloud computing as state-of-the-art mobile computing technology to alleviate the resource poverty of mobile devices by means of cloud-based resources. The solution supports a class of recommender systems that are context-aware, portable, accurate, and resource efficient.

Section 12.2 presents background details and the related research. Section 12.3 presents the proposed framework and its architecture. Section 12.4 presents the algorithms and tools to implement the framework. Section 12.5 presents the framework evaluation. Section 12.6 presents conclusions and future research.

12.2 Background and Related Research

First, we present the background details about the different types of recommender systems in Sect. 12.2.1. We then discuss the existing research on context-aware recommender systems in Sect. 12.2.2, e-type software recommender systems in

[1] Please note that we use the terms *proposed solution*, *proposed system*, and *proposed framework* interchangeably, all referring to the same concept.

Sect. 12.2.3, and cloud-based recommender systems in Sect. 12.2.4. A discussion of the different types of recommender systems and their state-of-the-art research helps us to justify the scope and needs for the proposed recommender system. The concepts and terminologies used in this section are utilized throughout the paper.

12.2.1 Types of Recommender Systems

The recommender systems can generally be categorized into five distinct types as illustrated in Fig. 12.2 that includes:

1. **Content-based recommender systems** recommend items to user according to user' preferences and past history [12].
2. **Collaborative filtering recommender systems** recommend the items to user based on the collaborative preferences that are gathered from a diverse set of users [44].
3. **Demographic recommender systems** recommend items to users on the basis of user's personal profile of demography [3].
4. **Knowledge-based recommender systems** recommend items according to either inferences regarding users' taste or particular domain knowledge. These types

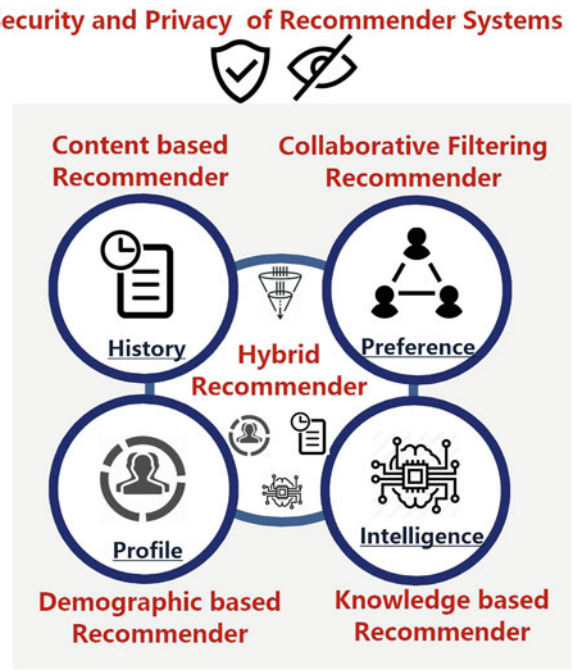


Fig. 12.2 An overview of the types of recommender systems

of recommender systems exploit past knowledge about how items of potential recommendation fit better according to preferences of the user [8].
5. **Hybrid recommender systems** are based on the intersection of the above mentioned approaches to provide recommendations [38].

In addition to the types of recommender systems mentioned above, we also discuss four major categories or the real world domains where different types of recommender systems have been applied [32]. We discuss the recommender systems in the context of mobile and e-type systems detailed below.

12.2.2 Context-Aware Recommender Systems

Context-aware recommender systems present items to the user(s) according to his/her taste and preferences as well as considering the contextual factors such as location, mood, weather, and day or time.

– *Mobile Recommender Systems for Places of Interest:* In recent years, there have been many efforts to develop recommender systems that can operate in different domains such as tourism, leisure, e-commerce, and mobile-commerce recommendations. For example, Braunhofer et al. [6] have developed a context-aware mobile application STS that suggests user's places of interests by using their mood, weather conditions, and personality traits. They used five factor model [15] along with user past rating to discover user personality traits. One of the primary challenge in developing multi-user mobile information systems lies with the scalability of the solution. By keeping scalability issues in mind, Roberts et al. [29] have designed a high-performance mobile recommender system (Magitti) which is scalable to many users operating at the same time. The system proposes a technique for recommending leisure time activities based on what time of week it is and venues nearby to the users' location by combining multiple recommendation patterns using predefined rules. A three-tier client–server architecture approach has been used to implement 3D based context-aware system [27]. These three tiers are the mobile client application, the GIS server, and the recommender server. The mobile client application is responsible to download and render 3D maps over cellular network. It also keeps track of user's location and speed via GPS/compass and communication is achieved through binary request–response protocol.
– *Knowledge-Based Recommender System for Movies:* Currently, recommender systems are making use of semantic web technologies to address the challenges of data sources diversity and information overload. For example, a recommendation system named as RecomMetz that recommends movie show times is presented in [8]. In RecomMetz, three different types of contextual factors are studied: *(a) location, (b) crowd, and (c) time* to produce recommendations. The proposed architecture of RecomMetz is based on modules, such as user interface, user check-in subsystem, data repository, context-aware subsystem, and recommendation engine.

Security and Privacy of Recommender Systems Typically, context-aware recommender systems have some privacy issues, such as the right(s) of users to know how, when, and under which circumstances their location as well as identity and other personal information can be accessible to other users or services. To support the privacy preserving mobile recommender systems, a solution named PRECISE [40] has been developed using cloud architecture. The PRECISE allows users to define privacy preserving policies while availing-off recommendation services.

12.2.3 Recommender Systems for E-Type Software

The e-type software refers to the systems such as e-health and e-commerce that exploit the ICT technologies to automate the manual and laborious tasks efficiently. Current recommendation techniques cannot be fully applied to e-type systems. To address this issue, Yang et al. [43] proposed a location-aware recommender system named PR (personal recommender) that fulfills customers' shopping demands with location-based seller offers and publicities. In this solution, on client side there are two components (a) web browser and (b) location manager. Customer requests include its GPS location, while the server side system maintains database for customer history and database having customer profile. With the help of customer past history which is stored in the form of customer preferences, similarity is estimated with any new web page by the vendor.

– *Recommendation System for E-Health:* Services related to health technology can be easily run over web due to current technological advancement in the field of cloud computing and strong infrastructure of wireless communication and sensor networks. Multiple organizations have provided online information regarding medical services available for public use. People make use of this information for personal health care management or patient-specific decision making [42].

  The information pertinent to the patients is generally distributed across a huge number of different web sites, so it is hard for patients to explore authentic health care information from large volume of data. It has been found that young people preferred to use mobile devices to download or browse health information [16]. Moreover, Wang et al. [41] have proposed a framework to develop a recommendation service that facilitates users to get relevant health information on mobile devices [35].

12.2.4 Cloud-Based Recommender Systems

Due to the limited battery and computational resources of mobile devices, cloud services provide a great alternative to software services that are configured and executed on the mobile. By taking into consideration the cloud computing based

Table 12.1 A comparison of the relevant existing solutions of mobile recommender systems

Application domain	Type of recommender	Context information	Mobile computing	Cloud computing	Mobile cloud computing	Solution reference
E-commerce	Content based	Location	✓	×	×	[43]
Leisure activities (music)	Content based, collaborative filtering	Weather, location, companion	✓	×	×	[14]
Leisure activities (media items)	Knowledge based, collaborative filtering	Location, time, day	✓	✓	✓	[26]
Leisure activities (movies)	Knowledge based, collaborative filtering	Location, crowd, time	✓	×	×	[8]
Tourism	Collaborative filtering	Location, time	✓	✓	✓	[20]
E-health	Collaborative filtering	User preferences, physiological data	✓	✓	✓	[41]
Tourism	Knowledge based, collaborative filtering	Location	✓	×	×	[25]

solutions, Otebolaku and Andrade [26] have proposed a context-aware recommendation system to recommend relevant cloud-based media particulars to mobile users. The context recognition service hosted inside the cloud is responsible for monitoring, learning, and predicting users' context. To retrieve user context, WiFi, GPS, accelerometer, rotation as well as orientation vector sensors have been used. In this solution, the nearest neighbor (KNN) algorithm has been used to predict the location and activity of the users. Moreover, a web-based recommender system named REJA is developed to address drawbacks of mobile recommender systems [25]. The current approaches detailed above do not provide an optimal solution for the problem of group recommendations as well as cold start and data sparseness problems. To overcome these issues, Khalid et al. [20] have implemented a cloud-based solution named OmniSuggest for the problem of venue recommendation in the domain of social networks for a single user and/or a group of friends. OmniSuggest uses the mixture of ant colony algorithms with social filtering technique to retrieve the most favorable location recommendations based on real-time context such as traffic and weather conditions.

Comparative Summary of Existing Solutions In Table 12.1, we present a comparison-based summary of the existing solutions. For an objective comparison and interpretation of the results, we compare the existing solutions based on six distinct criteria presented in Table 12.1. For example, to interpret the data in Table 12.1, we can summarize that [43] presents a solution that exploits the context-based recommender techniques and uses location as a context to support mobile recommender system for e-commerce activities. We conclude that the solutions for mobile recommender systems have progressed and matured over time. However, there is a need for innovative solutions that address mobile commerce in general and smart markets in particular. In addition, the emerging solutions need to exploit mobile cloud computing as state-of-the-art for mobile computing technology. Mobile cloud computing supports context-aware and mobility-driven recommendations while also maintaining the scalability and elasticity of computational resources.

12.3 Architecture of the Recommendation Framework

In this section, we present the architecture of the proposed framework and its underlying layers that represents a higher-level view and blueprint of the overall system. Based on the presented architecture, we discuss the implementation specific details for the framework later in the paper.

12.3.1 Architecture and Patterns for the Framework

As per the ISO/IEC/IEEE 42010 standard,[2] architecture of a software intensive systems represents a high-level view of the systems in terms of system components (e.g., computational elements and data stores) and connectors that enable component communication. We follow software architecture-based development of the proposed framework for two reasons detailed below.

1. **System abstraction and quality:** Software architecture abstracts the complex and implementation specific details of the system with higher-level software components and connectors. Software components and connectors help with designing and reasoning about the system functionality and quality prior to its implementation [23].
2. **Reusability of design:** Software architecture patterns can be exploited as proven best practices that support reusability of components and structures in architecture-based development and enhance the quality of the software [7].

[2]ISO/IEC/IEEE 42010 Systems and software engineering—Architecture description is an international standard for architecture descriptions of systems and software.

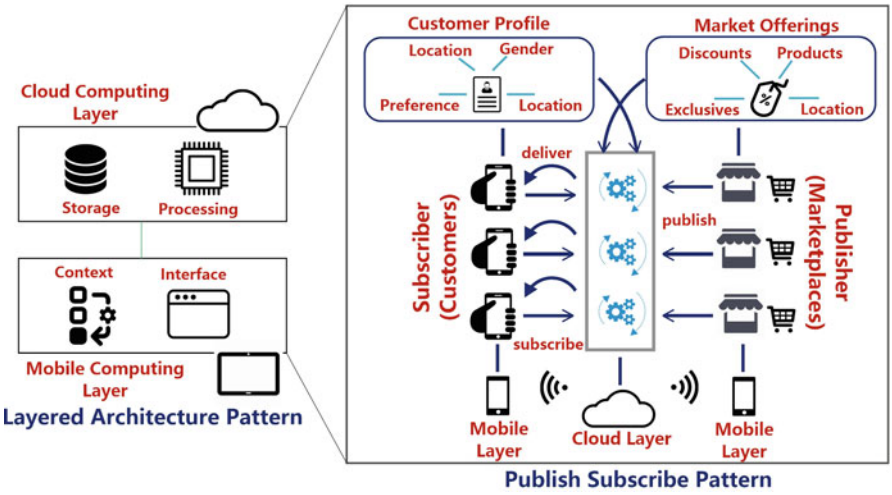


Fig. 12.3 Pattern-based software architecture for recommender system

We present the architecture of the proposed framework in Fig. 12.3. Specifically, we present the software architecture view in terms of two architectural patterns, namely: *layered architecture pattern*, and *publish–subscribe pattern* as illustrated in Fig. 12.3. The layered architecture pattern has helped us to maintain the separation of concerns in terms of mobile and cloud computing layers to engineer and develop the recommender system[18]. In addition, we have also applied the publish–subscribe pattern that helps to maintain the relationship between potential customers and business entities as the requesters and the providers of the product specific contextual information [13]. Based on the illustration in Fig. 12.3, we detail the application of the layered pattern to the proposed software architecture as below.

12.3.2 Context-Aware Mobile Computing Layer

The mobile computing layer as the front-end of the system exploits portable and context-aware mobile devices that provides an interactive interface to the potential customers and the market entities to communicate with the system [2, 5]. Using the mobile computing layer the potential customers can specify their preferences such as interest in available discounts, consumer products, and services to enable the matchmaking between potential customers and the market. In addition to the user input and decision support, the mobile device dynamically calculates the contextual information such as user's geographical location along with market offerings to recommend the products/services of the customers' interest.

Mobility of mobile computing inherits a few challenges such as resource poverty that includes limited processor, memory, and available energy to perform complex

and computationally intensive tasks. This restriction poses the challenge to the mobile recommender system that must perform complex analytics—as real-time computations—to offer accurate recommendations. There is a need to extend the computation and storage resources of the mobile devices to enable efficient and scalable recommender system.

12.3.3 Computation-Based Cloud Computing Layer

Cloud computing layer as the back-end of the system provides virtually unlimited (pay-per-use) hardware and software resources [24]. Specifically, the cloud layer offers infrastructure, platform, and software as a service to its users. Therefore, in order to compensate for the resource poverty of the context-aware mobile device, we use the software as a service offered by cloud servers that integrates the mobile and cloud computing technologies to generate the recommendations.

Once the mobile device captures user preferences and contextual information, the details are stored on the cloud-based server. The cloud server based on the input from the mobile computing layer computes the most relevant recommendations and communicates them back to the mobile device. The integration and operations of the mobile and cloud computing technologies are enabled by means of continuous availability of the network that enables the inter-layers communication. By using the layered architecture pattern, we distinguish between the two distinct concerns of user interaction and system processing with a systematic implementation of the recommendation system.

As in Fig. 12.3, the publish–subscribe patterns help to manage an effective coordination between the user level inputs and the system level processing. By applying this pattern, the market entities (such as outlets and restaurants) can publish their offerings of products/services to a central repository (such as cloud-based data storage) for their broadcasting. In contrast, the potential customers can subscribe to the published offerings that enable the matchmaking between both parties. The publish–subscribe patterns provide a systematic mediation between markets and potential customers to enable the digital matchmaking.

12.4 Algorithms and Technologies for Framework Implementation

After presenting the software architecture, we now present the details of the architecture-based implementation of the recommender framework. We present the algorithms that represent the data, modularization, and parameterized customization of the proposed framework in Sect. 12.4.1. We discuss the tools and technologies that implement the algorithms to provide the automation and proof-of-the-concept for the recommender system in Sect. 12.4.2.

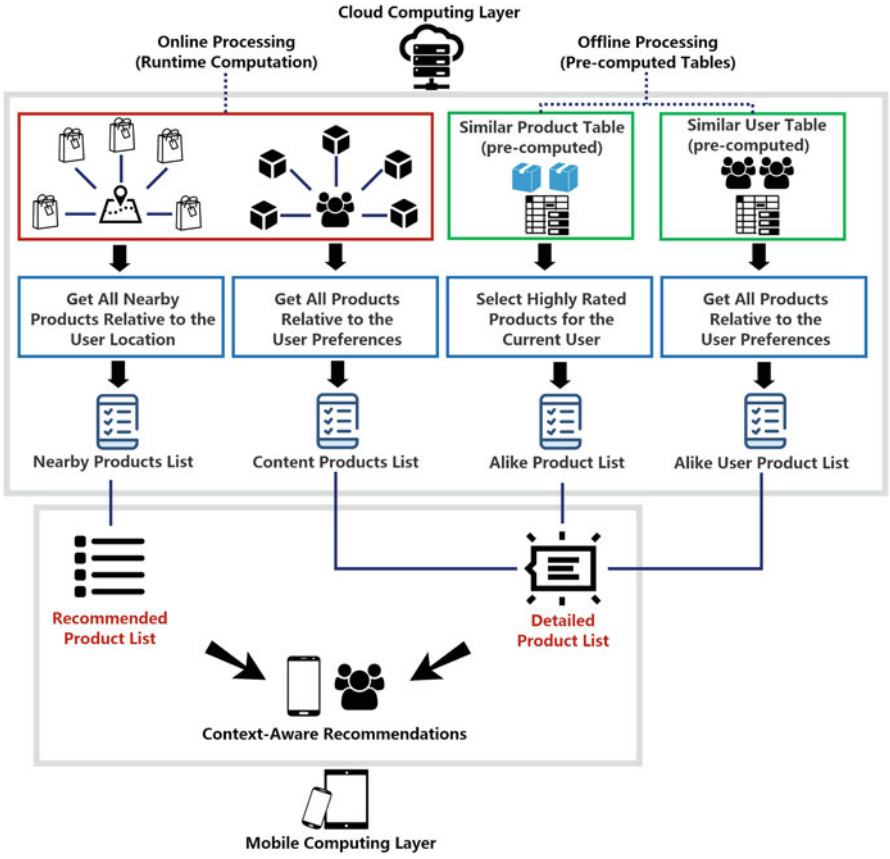


Fig. 12.4 An overview of the context-aware recommendation algorithms

12.4.1 Algorithms for Recommender System

We present the algorithms for the recommender system guided by the illustrations in Fig. 12.4. First the *recommend-products* algorithm is executed to collect the details of the nearby products (as per users' location and proximity details). The recommend-products algorithm is referred to as the online processing as it dynamically calculates the relevant products and offers based on users' location and preferences each time the recommender system is executed. The next two algorithms *similar-users-product-ratings* and *similar-product-ratings* support complementary functionality to find similar products and users. These two algorithms are precomputed, normally as part of the off-line processing. The technical details of these algorithms as in Fig. 12.4 are provided below (Table 12.2).

In the context of the existing recommendation systems (cf. Sect. 12.2, Fig. 12.2) We have adopted the hybrid recommendation system approach. This approach uti-

Table 12.2 Utility methods of algorithms to generate context-aware recommendations

Method (parameter)	Returns	Description
FIND-SIMILAR-USERS (id)	List	Get products recommended to similar users having high scores corresponding to active user
FIND-SIMILAR-PRODUCTS (user id)	List	List products from similar-products table having high scores corresponding to active user
GET-LOCAL-PRODUCTS (user location)	List	List all available products near to active user's current location
GET-PRODUCT-PREFERENCES (user id)	List	List all available products based on active user's preferences
GET-TOP-PRODUCTS (similar product list)	List	Get list of top k products from similar-products table rated highly corresponding to active user
GET-TOP-USERS-PRODUCTS (similar user list)	List	Get list of top k products from similar-users table rated highly corresponding to active user

Table 12.3 Parameters of algorithms to generate context-aware recommendation

Parameter	Description
P_{local}	List of products located in the vicinity of the active user's current location
$P_{preferences}$	List of products falling in similar category of active user's preferences
$P_{similar}$	List of products from similar products list rated highly by the active user
$P_{similar-user}$	List of products from similar-users table rated highly for the active user
$P_{detailed}$	Combined product list of three lists that includes "content product list," "alike product list," and "alike user-product list"
$P_{recommended}$	Contains common products in both detailed and nearby product lists

lizes the context information along with content filtering technique and collaborative filtering algorithm to generate best possible recommendations. The utility function of the implemented system is as follows:

$$\text{Context} \times \text{User} \times \text{Product} \rightarrow \text{Recommendation}$$

All the algorithmic execution and data storage take place at the back-end cloud server. Mobile devices only act as portable and context-aware user interfaces to provide some input (user location and preferences) and output (context-aware recommendations) as in Fig. 12.4. The unification of the mobile and cloud computing helps with a portable, context-aware recommender system with necessary computation and storage resources.

Table 12.3 presents a list of variables that support the parameterization of algorithms for online recommendation's generation. Also, Table 12.2 highlights all the utility methods that are used during the process of online recommendation generation.

Algorithm 1: Recommend-Products

- **Input:** Active user's id (uid), geolocation of the user (loc), and preferences of the user (preferences).
- **Process:** Based on the active user's location all locally available products and offerings are selected and compiled as a list (P_{local}—Line 2, Algorithm 1). This list is used to retrieve a detailed product list as per the user preferences ($P_{detailed}$—Line 4). A procedure runs to find similar products using preferences of similar users. Once all duplicated data is removed, a recommendation is returned based on top k items that rated highly for the active user ($P_{recommended}$—Line 5) in Algorithm 1. The tables comprising similar users and products are precomputed using off-line algorithms described later.
- **Output:** A list of recommended products $P_{recommended}$.

Algorithm 1 Recommend-products algorithm

Require: current user: uid, geolocation: loc, user preferences: $P_{preference}$
1: $P_{local} \leftarrow$ GET-LOCAL-PRODUCTS(loc)
2: $P_{preference} \leftarrow$ GET-PRODUCT-PREFERENCES(uid)
3: $P_{similar} \leftarrow$ FIND-SIMILAR-PRODUCTS(uid)
4: $P_{detailed} \leftarrow P_{local} \cap (P_{preference} \cup P_{similar})$
5: $P_{recommended} \leftarrow$ GET-TOP-PRODUCTS(P)
Ensure: $P_{recommended}$ a list of recommended products

Algorithm 2: Similar-Users-Product-Ratings

- **Input:** User-product rating matrix ($\mathcal{U}sers$).
- **Process:** The process illustrated in Algorithm 2 picks all products that are not rated by the current user (ρ—Line 3, Algorithm 2). In the next step, a locality-based criterion based on the user location is used to find all neighboring users with available product ratings ($P_{similar-user}$—Line 4). Only users within the same demographic category are considered while compiling the neighboring users' set. In the end, ratings for the target product are calculated based on a weighted average of neighboring users' rating ($P_{predicted}$—Line 5). The resulting ratings are updated to the user-product rating matrix.
- **Output:** A list of predicted product ratings $P_{predicted}$.

Algorithm 3: Similar-Product-Ratings Algorithm

- **Input:** Product rating matrix ($\mathcal{P}roducts$).
- **Process:** First all similar users who have not rated a target product are selected ($U_{similar}$—Line 1, Algorithm 3). In the next step, all other products rated by those users are searched. Cosine similarity is used to calculate product similarity

Algorithm 2 Similar-users-product-ratings algorithm

Require: user-product rating matrix: ($\mathcal{U}sers$)
1: Initialize $V(s) = 0$, for all $s \in \mathcal{S}^+$
for each: $u \in \mathcal{U}sers$
2: $P_{user} \leftarrow$ GET-PRODUCTS(u)
3: $U_{similar} \leftarrow$ GET-SIMILAR-USERS(P_{user})
4: $\rho \leftarrow$ PEARSON($u, U_{similar}$)
5: $P_{similar-user} \leftarrow$ GET-SIMILAR-PRODUCTS(ρ)
6: $P_{predicted} \leftarrow$ PREDICT-PRODUCTS($P_{similar-user}, u$)
Ensure: $P_{predicted}$ a list of predicted product ratings

of the target product to other products (ρ—Line 3). This results in a subset of most similar products ($P_{similar}$—Line 4). Subsequently, a rating for the target product is predicted using the ratings of the similar products. The predicted rating is updated against the target product in the user-product rating matrix ($P_{predicted}$—Line 4).

– **Output:** A list of predicted product ratings $P_{predicted}$.

Algorithm 3 Similar-product-ratings algorithm

Require: user-product rating-matrix
for each: $p \in \mathcal{P}roducts$
1: $U_{similar} \leftarrow$ GET-SIMILAR-USERS(p)
2: $P_{similar} \leftarrow$ GET-SIMILAR-PRODUCTS(U, p)
3: $\rho \leftarrow$ COSINE(p, P)
4: $P_{similar} \leftarrow$ GET-SIMILAR-PRODUCTS(ρ)
5: $P_{predicted} \leftarrow$ PREDICT-RATING($P_{similar}$)
Ensure: $P_{predicted}$ a list of predicted product ratings

12.4.2 Tools and Technologies for Framework Implementation

After presenting the algorithms, we now discuss the tools and technologies used to implement the framework. The framework implementation represents a prototype based proof-of-the-concept for the proposed solution. We have used the architecture from Fig. 12.3 to implement the framework. An overview of the integrated tools and technologies to implement the mobile computing and cloud computing layers is provided in Fig. 12.5.

We have exploited the Amazon cloud services for storage and computing efficiency. From a technical perspective, we have deployed a virtual server on Amazon cloud called Amazon EC2 instance[3] and set up Red Hat Linux operating system over that instance. We have developed server side application using Node.js[4]

[3] Amazon EC2: https://aws.amazon.com/ec2/.

[4] Node.js: https://nodejs.org/en/.

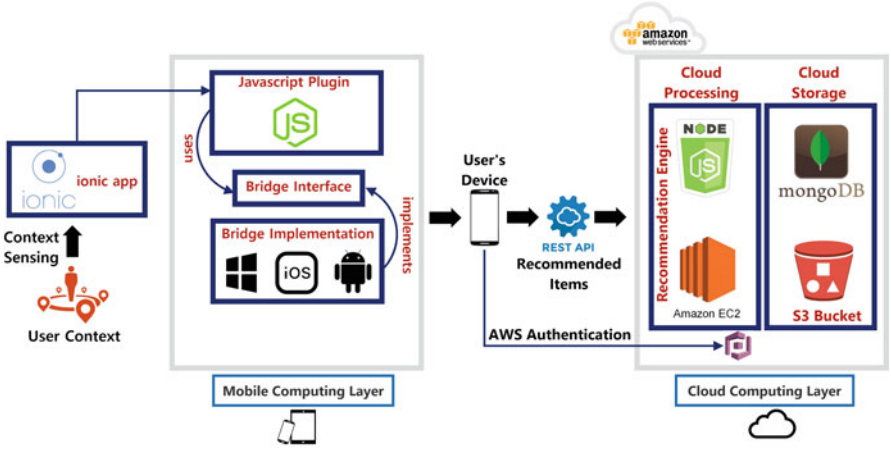


Fig. 12.5 Overview of the tools and technologies to implement framework

and set up Node.js web server on Amazon EC2 Instance. For the sake of efficient data retrieval we have used Mongo DB.[5] We installed MongoDB on Amazon EC2 Instance. Recommendation related data is managed by Amazon S3 storage services.[6]

The recommendation engine is written in python language[7] installed over Amazon EC2. It is the main component of the framework. The task of recommendation engine is to retrieve data from MongoDB collections, run recommendation algorithms and techniques, and save the result set back to MongoDB for user's recommendations. The algorithmic details have already been discussed in the previous section.

12.4.3 Implementing Context-Aware Mobile Computing Layer

From an implementation point of view, at the mobile computing layers (i.e., context-aware user interface) we have exploited HTML5[8] technologies to support multiple mobile platforms. Moreover, the reason to choose a platform independent technology for the mobile layer is that the framework carries out all performance intensive tasks over cloud layer. The core responsibility of the mobile layer is to retrieve the current location of the user and calculate users' preferences to (1)

[5]MongoDB: https://www.mongodb.com/.
[6]Amazon S3: https://aws.amazon.com/s3/.
[7]Python: https://www.python.org/.
[8]HTML 5—World Wide Web Consortium: https://www.w3.org/TR/html5/.

send them to cloud end of the system, and (2) display the recommended products to the end user. To find the current location of the user we have used HTML5 Geolocation API. Restful Architecture has been utilized to perform communication between mobile and cloud end of the system. The user of the framework needs to be registered to system to get any recommendations. User information is stored into user table/collection in MongoDB installed over Amazon EC2. After the user gets logged into the system, user's current location is retrieved. The current location of the active user and preferences are sent to Node JS server via Restful API.

12.4.4 Implementing Processing Based Cloud Computing Layer

This section describes the methodological details to implement cloud end of the recommender system. It is vital to mention that all computational and storage work for recommendation generation is performed over cloud layer using node JS server and MongoDB that are deployed on Amazon EC2 instance. Another benefit gained by cloud layer is off-line processing performed by python based recommendation engine. The purpose of off-line processing is to overcome the problems of scalability and performance.

To implement off-line processing we have used collaborative filtering algorithm [8, 25, 44]. We have utilized the table user-product rating matrix to apply above mentioned techniques and generate precomputed tables of similar users as well as similar products (cf. Algorithms 2 and 3). Python based recommendation engine runs these algorithms to refresh precomputed tables on daily basis based on the time when there is a minimalistic use of the framework. In order to compute item based similarities we have utilized cosine-based similarity (cf. Algorithm 3—Line 3) and for the sake of user based similarities' computation, we have utilized Pearson correlation method (cf. Algorithm 2—Line 3). We have unified the items based collaborative filtering technique with user based collaborative filtering technique during off-line processing.

Figure 12.6 presents an overview of the context-based recommendation of the products' list to the users. As highlighted in Fig. 12.6, there are two types of users, namely: (1) new users and (2) existing users that lead to two scenarios for the recommendations that are detailed below.

- **Cold Start Scenario** represents the situation when a new user utilizes the framework for context-aware recommendations as illustrated in Fig. 12.6a. Since the framework has no prior information about the user's context, any computations and recommendations by the framework are cold start. In the cold start scenario, the framework gathers user's location and preferences to generate the recommendations.
- **Warm Start Scenario** represents the situation when an existing user utilizes the framework to get the recommendations as illustrated in Fig. 12.6b. In this

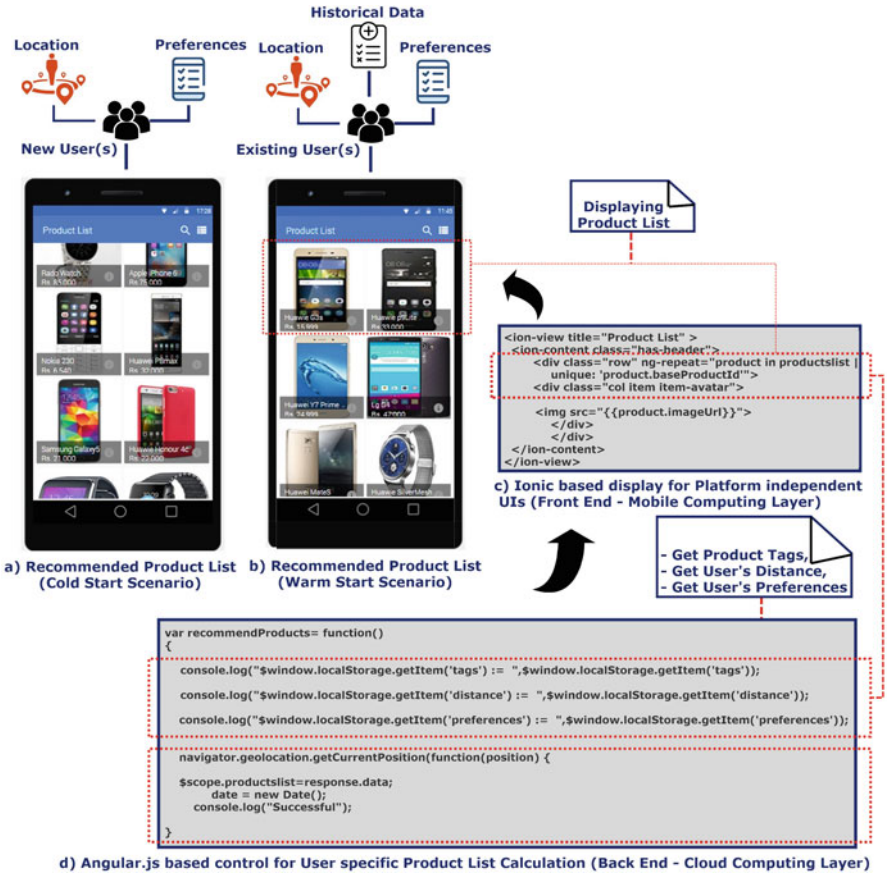


Fig. 12.6 Overview of the product recommendations sample implementation logic

situation, the framework has prior contextual information about the user that helps the framework's accuracy of recommendations as a warm start. In the warm start scenario, the framework gathers user's location, preferences as well as user's historical data (e.g., past preferences, items/points of interests, and time/date) to generate the context-aware recommendations. These recommendations ensure a digital matchmaking in a virtualized context of smart markets involving potential customers and business entities.

We also highlight the sample code that executes at the mobile and cloud computing layers of the framework to generate the context-aware recommendations in Fig. 12.6. Figure 12.6c presents the partial view of the code to display the recommender product list to the users. For the display of the recommendation list, we have used ionic framework to support a platform independent code/technology. Figure 12.6d presents the back-end logic that gathers the user's contextual informa-

tion to generate the recommendation list. The logic is executed at the cloud-based server. We have used the Angular JS to compute the contextual information.

12.5 Qualitative Evaluation of the Framework

We now present the results for the framework evaluation. Specifically, we discuss the dataset(s) used in Sect. 12.5.1 to evaluate the accuracy and efficiency of the framework presented in Sect. 12.5.2. Finally, we also present some threats to the validity of the evaluation results detailed in Sect. 12.5.3.

12.5.1 Platform, Metrics, and Dataset for Evaluation

- **Platform and Tools Used for Evaluation:** All evaluations were performed using Huawei P8 Lite smart-phone on the client side (i.e., mobile computing layer). On the other hand, on the server side, a Red Hat Enterprise Linux OS system with Node.js, MongoDB, and Python installed was used (i.e., cloud computing layer). The proposed recommender framework is evaluated based on two main criteria: (1) accuracy of the recommendations that are generated by the cloud server and (2) efficiency of the resource utilization by the mobile device in terms of CPU and power consumption. Memory consumption issues are not considered due to the fact that all the data storage takes place at the cloud-based server.
- **Metrics Used for Evaluation:** To evaluate software quality features of the developed recommender system, we have used ISO/IEC 9126-1 software quality standard [19] introduced by the International Organization for Standardization (ISO).[9] The standardized model investigates six quality features that are categorized into 27 sub-categories. Using an established model to evaluate the quality of the framework can help us to avoid any bias and guides feature based evaluation of the framework. To evaluate the proposed recommender framework, we only considered two quality features: *accuracy*, *efficiency*, and their sub-features.
- **Dataset Used for Framework Evaluation:** We used the dataset of superstore sales to evaluate the proposed recommender system. The dataset contains real items of product offerings and is publicly available at [30]. The selected dataset provides us with realistic data and scenarios to avoid any bias or limitations of the evaluation. We slightly modified the dataset to accommodate geographic locations (regions/provinces) corresponding to our needs that provides the foundation to evaluate the framework in a real context, as per the needs of the

[9]It is noteworthy that ISO/IEC 9126–1 was first published in 1991; and later on from the year 2001 to the year 2004 ISO published an international standard (ISO/IEC 9126–1) as well as three technical reports (ISO/IEC 9126–2 to ISO/IEC 9126–4).

local users and markets. For framework evaluation, we consider attributes of *customer name*, *product name*, *product category* and *sub-category*, *price*, and *geographic location* to propose a recommender system.

12.5.2 Results for Framework Evaluation

The result shows higher precision rates for the proposed system corresponding to good recommendations. Moreover, the proposed system demonstrated an efficient CPU consumption and memory usage.

Accuracy of Framework's Recommendations

In order to quantify and measure the accuracy of the framework's recommendations, we have used two metrics, namely: (a) *recommendation precision* as a measure of the accuracy of the recommendations made, and (b) *recommendation recall* measures the proportion of correct recommendations out of the total recommendations made by the framework. Mathematically, the metrics are defined as

$$\text{precision} = \frac{TP}{TP + FP} \; ; \text{recall} = \frac{TP}{TP + FN}$$

where

$$\begin{cases} TP: \text{ true positives} \\ FP: \text{ false positives} \\ TN: \text{ true negatives} \\ FN: \text{ false negatives} \end{cases}$$

We present the results of measuring the framework's accuracy based on the data in Table 12.4. Moreover, we provide an illustrative comparison of the evaluations and trials on the framework to measure its accuracy in Fig. 12.7. As highlighted in Table 12.4 and Fig. 12.7, there are two scenarios, namely: cold start and warm start recommendations that have been detailed earlier.

We performed the trials with 5 distinct user groups (UG), where each group had on average 10 people with varying age and gender groups along with distinct preferences to reduce any bias in the recommendation trials. The users' groups were asked to specify their preferences and let the framework provide them with context-aware recommendations. Based on the data in Table 12.4 and its visualization in Fig. 12.7, we observed that average precision and recall of the proposed system were 80.4% and 64.8%, respectively, in case of newly registered users (cold start scenarios). On the other hand, average precision and recall of the system were 82.6% and 72.6%, respectively, in case of already existing users (warm start scenarios).

Table 12.4 Precision and recall for newly registered users and existing user

	User group	Specific category total	Recommended	Relevant	Precision	Recall
Cold start scenario	UG1	81	63	45	71%	55%
	UG2	96	72	60	83%	62%
	UG3	98	98	84	85%	85%
	UG4	90	60	50	83%	55%
	UG5	91	65	52	80%	57%
	Avg.	91.2	71.6	58.2	80.4%	64.8%
Warm start scenario	UG1	84	96	72	75%	85%
	UG2	99	77	66	85%	66%
	UG3	72	63	54	85%	75%
	UG4	88	66	55	83%	62%
	UG5	80	70	60	85%	75%
	Avg.	84.6	74.4	61.4	82.6%	72.6%

Note: Column "specific category total" shows the total number of records related to a particular topic in the database. Column "recommended" represents the count of retrieved records, while column "relevant" represents the number of records relevant to user preferences. The last row in both tables describes the average for each column

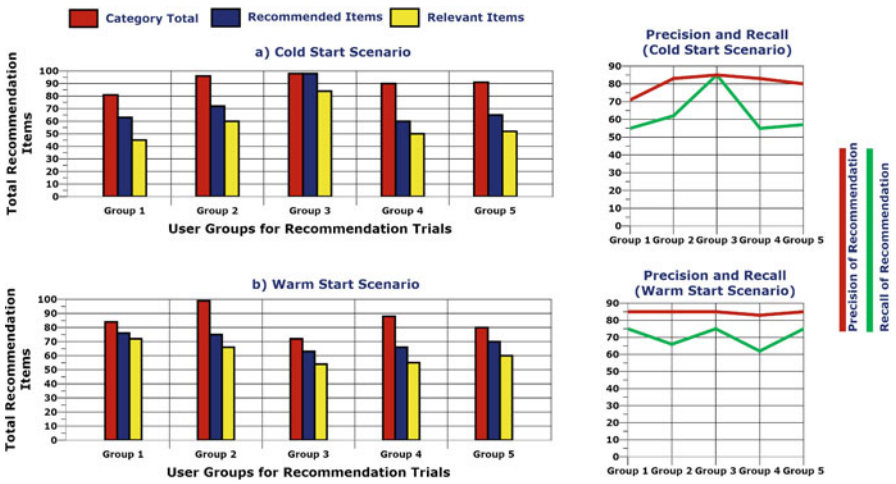


Fig. 12.7 Overview of the results for evaluating framework's accuracy for recommendation

A recommendation list was generated based on the location of the user as in Fig. 12.6. The recommendation list was also used to record and evaluate the precision and recall of recommendations provided by the system. The results have been shown in Table 12.4 for newly registered users and existing users, respectively.

Efficiency of the Framework

We now measure the computational and energy efficiency of the proposed recommender system. The data is offloaded to cloud-based server, therefore, evaluating the memory or storage efficiency of the mobile computing layer is out of the scope here. To assess and evaluate the efficiency of the recommender framework, we monitored its memory and CPU usage using CPU monitor [9]. An overview of the framework's processing and power efficiency monitoring is illustrated in Fig. 12.8. In Fig. 12.8, to measure the efficiency, we need to consider two execution modes of the framework:

- **Framework Execution as a Foreground App** represents the scenario when the framework is active and fully executional during the recommendations process.
- **Framework Execution as a Background App** represents the scenario when the framework is only running in the background for context calculation but not operational for the users' recommendations.

We observed that the proposed application took approximately 3 s to fetch and display a list of recommended items acquired from the node server to the end user. In the former mode, CPU consumption did not exceed 4% and RAM used was 157 MB. In the latter mode, maximum CPU usage remained 2% and RAM usage was ∼146 MB. Furthermore, to measure battery consumption of the mobile application, we used AccuBattery [1]. The usage was normal in both execution modes, 0.2% and 0.4% in background and foreground modes, respectively.

12.5.3 Threats to the Validity of Framework

After presenting the framework evaluation, we also highlight some threats to the validity of the proposed research and solution. The threats also highlight the possible future work to optimize the proposed solution.

- **Threat I—Availability of the Diverse Dataset:** A possible threat to the validity relates to the availability of a diverse set of data. Diversity of data refers comprehensiveness of the user related information (i.e., gender, social and national background, emotion, etc.) along with time, day, and other environmental conditions to further evaluate the framework. The proposed algorithms provide parameterized customization of the solution. However, the availability of the diverse dataset can help us to further evaluate the accuracy and efficiency of the proposed and developed framework.
- **Threat II—Real Use-Cases from Smart Markets:** Smart recommender systems in general and smart markets in particular are relatively innovative concepts and lack any historical data. Unlike the more conventional recommender systems, the available usage scenarios for smart markets are limited. Moreover, the unification of the mobile and cloud computing technologies requires historical data and use-case for a more rigorous evaluation of the framework.

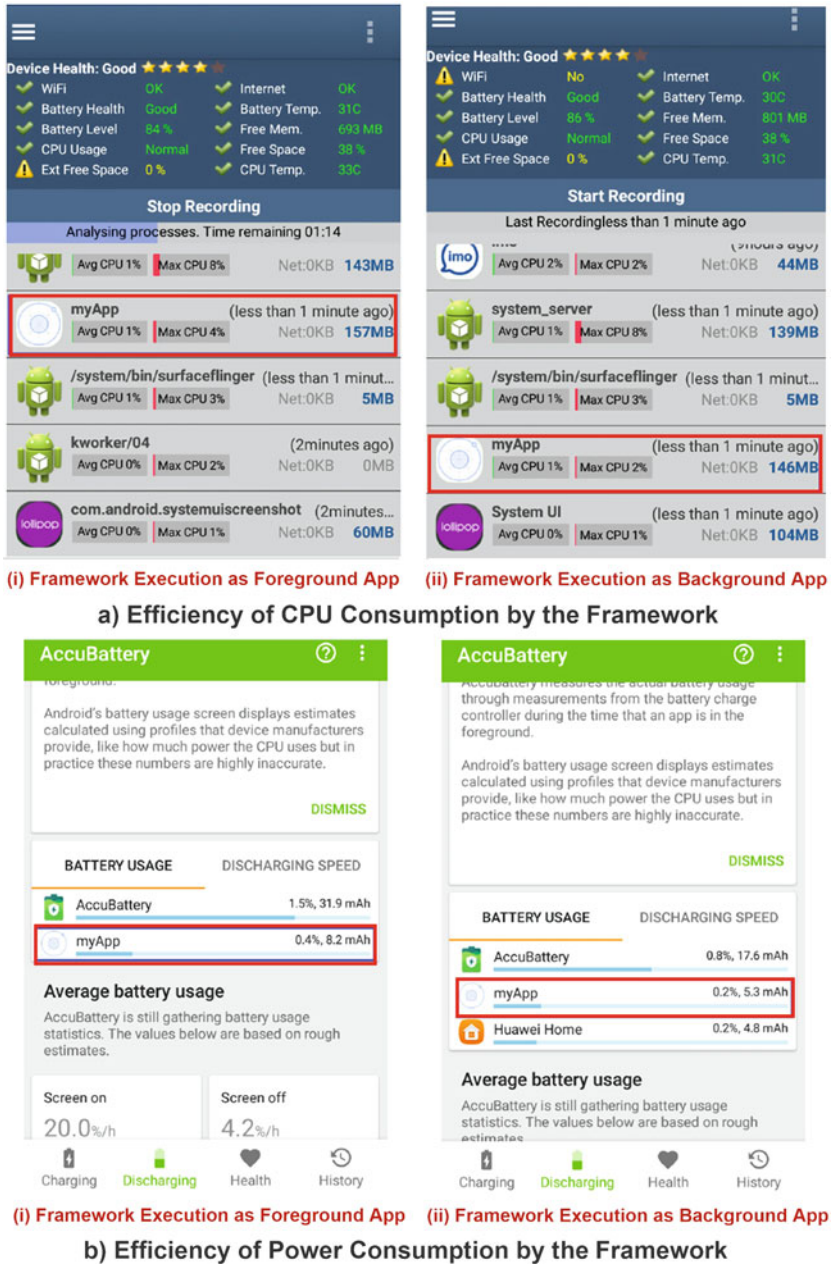


Fig. 12.8 Overview of the results for evaluating framework's efficiency

– **Threat III—24 × 7 Connectivity for Framework:** From a technical perspective, the solution exploits mobile devices as context-sensitive and portable user interface. The resource poverty of the mobile devices is alleviated with a continuous connectivity and processing at the cloud-based server. Therefore, a fundamental requirement to the success of the framework is a continuous network connection between a mobile device and the cloud-based server. In case of poor connectivity, the accuracy and performance of the framework can be affected.

12.6 Conclusions and Future Research

In this paper, we exploit the mobile cloud computing as state-of-the-art mobile computing technology to develop a context-sensitive and portable recommender system. The recommender system aims to support an efficient and context-driven matchmaking between potential customers (based on their shopping preferences) and relevant business entities (based on their products/service offerings). The recommender system supports the activities of the smart markets, i.e., virtualized and context-aware markets and/or shopping arena. Smart markets aim to facilitate the customers with recommendations and supporting business entities to maximize the outreach of their products and offerings. The proposed recommender system advances the state-of-the-art for recommender systems by exploiting a layered architecture that unifies the mobile computing and cloud computing technology layers. The system supports smart city systems in general and focuses specifically on smart markets.

Contributions and Outcomes The primary contributions of the solution lies with the proposed architecture and its underlying algorithms to sense contextual information from the users. The contextual information is matched with the best market offerings to facilitate the users with decision support based on contextual recommendations. We have used the publish–subscribe architectural pattern to reuse knowledge and best practices and customized it to enable an effective matchmaking and communication. The architectural model and pattern used have also helped us to model and develop mobile computing layer (front-end context-sensitive user interface) that relies on cloud computing layer (back-end data processor and storage) to alleviate the resource poverty of the mobile devices. In short, the proposed research presents an architecture, patterns, algorithms, enabling technologies, and implementation platform to develop a recommender system for smart markets.

Evaluations and Limitations We have developed and evaluated the prototype as a proof-of-the-concept for recommender system and its underlying algorithms. The prototype supports automation, user intervention, and customization during the recommendation process. We have used the ISO/IEC-9126-1 model to evaluate the quality in terms of accuracy and efficiency of the recommender system. To support a formal approach, we have utilized cosine-based similarity and Pearson correlation

for computation of context-aware recommendations. The evaluation results suggest that the framework supports high accuracy for recommendations and facilitates computation and energy efficient mobile computing. We have also highlighted some threats to the validity of the research.

Future Work In future, we mainly focus on extending the types of recommendations and its application to other domains of the smart city systems such as crowd-sensed recommendations. Also, the privacy of user's context, their preferences, and information are also of central importance as part of the future work. From the functional perspective, we aim to explore other contextual factors, such as weather conditions, a week of the day, etc., to further optimize and extend the types of recommendations.

References

1. Accbattery(version-1.1.7). https://play.google.com/store/apps
2. Ali, M., Zain, J.M., Zolkipli M.F., Badshah, G.: Mobile cloud computing & mobile battery augmentation techniques: a survey. In: 2014 IEEE Student Conference on Research and Development (SCOReD), pp. 1–6. IEEE, Piscataway (2014)
3. Al-Shamri, M.Y.H.: User profiling approaches for demographic recommender systems. Knowl.-Based Syst. **100**, 175–187 (2016)
4. Bakıcı, T., Almirall, E., Wareham, J.: A smart city initiative: the case of Barcelona. J. Knowl. Econ. **4**(2), 135–148 (Jun 2013)
5. Baltrunas, L., Ludwig, B., Peer, S., Ricci, F.: Context relevance assessment and exploitation in mobile recommender systems. Pers. Ubiquit. Comput. **16**(5), 507–526 (2012)
6. Braunhofer, M., Elahi, M., Ricci, F.: Sts: a context-aware mobile recommender system for places of interest. In: UMAP Workshops. Citeseer, 2014.
7. Buschmann, F., Henney, K., Schimdt, D.: Pattern-Oriented Software Architecture: On Patterns and Pattern Language, Vol. 5. John Wiley & Sons, Hoboken (2007)
8. Colombo-Mendoza, L.O., Valencia-Garcia, R., Rodriguez-Gonzalez, A., Alor-Hernandez, G., Samper-Zapater, J.J.: Recommetz: a context-aware knowledge-based mobile recommender system for movie showtimes. Expert Syst. Appl. **42**(3), 1202–1222 (2015)
9. Cpumonitor(version-6.54). https://play.google.com/store/apps
10. Dameri, R.P.: Smart City and Digital City Implementation: Two Best Practices in Europe, pp. 109–154. Springer, Berlin (2017)
11. da Silva, W.M., Alvaro, A., Tomas, G.H.R.P., Afonso, R.A., Dias, K.L., Garcia, V.C.: Smart cities software architectures: a survey. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing, pp. 1722–1727. ACM, New York (2013)
12. de Gemmis, M., Lops, P., Musto, C., Narducci, F., Semeraro, G.: Semantics-aware content-based recommender systems. In: Recommender Systems Handbook, pp. 119–159. Springer, Boston (2015)
13. Demers, A., Gehrke, J., Hong, M., Riedewald, M., Walker, W.: Towards expressive publish/-subscribe systems. In: EDBT, vol. 6, pp. 627–644. Springer, Berlin (2006)
14. Derwein, C., Beer, W., Hargassner, W., Herramhof, S.: General Framework for Context-Aware Recommendation of Social Events. IARIA, Vienna (2013)
15. Gosling, S.D., Rentfrow, P.J., Swann, W.B.: A very brief measure of the big-five personality domains. J. Res. Pers. **37**(6), 504–528 (2003)
16. Hasman, L.: An introduction to consumer health apps for the iphone. J. Consum. Health Internet **15**(4), 322–329 (2011)

17. Jamshidi, P., Ahmad, A., Pahl, C.: Cloud migration research: a systematic review. IEEE Trans. Cloud Comput. **1**(2), 142–157 (2013)
18. Jones, N.C., Meter, R.V., Fowler, A.G., McMahon, P.L., Kim, J., Ladd, T.D., Yamamoto, Y.: Layered architecture for quantum computing. Phys. Rev. X **2**(3), 031007 (2012)
19. Jung, H.-W., Kim, S.-G., Chung, C.-S.: Measuring software product quality: a survey of ISO/IEC 9126. IEEE Softw. **21**(5), 88–92 (2004)
20. Khalid, O., Khan, M.U.S., Khan, S.U., Zomaya, A.Y.: OmniSuggest: a ubiquitous cloud-based context-aware recommendation system for mobile social networks. IEEE Trans. Serv. Comput. **7**(3), 401–414 (2014)
21. Kitanov, S., Janevski, T.: State of the art: mobile cloud computing. In: 2014 Sixth International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN), pp. 153–158. IEEE, Piscataway (2014)
22. Lewis, G.A., Lago, P., Procaccianti, G.: Architecture strategies for cyber-foraging: preliminary results from a systematic literature review. In: European Conference on Software Architecture, pp. 154–169. Springer, Berlin (2014)
23. Medvidovic, N., Taylor, R.N.: Software architecture: foundations, theory, and practice. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, vol. 2, pp. 471–472. ACM, New York (2010)
24. Mell, P., Grance, T.: The NIST definition of cloud computing. Commun. ACM **53**(6), 50 (2010)
25. Noguera, J.M., Barranco, M.J., Segura, R.J., MartíNez, L.: A mobile 3D-GIS hybrid recommender system for tourism. Inf. Sci. **215**, 37–52 (2012)
26. Otebolaku, A.M., Andrade, M.T.: Supporting context-aware cloud-based media recommendations for smartphones. In: 2014 2nd IEEE International Conference Mobile Cloud Computing, Services, and Engineering (MobileCloud), pp. 109–116. IEEE, Piscataway (2014)
27. Postel, J.: RFC 793: transmission control protocol, September 1981. Status: Standard **88** (2003)
28. Rappaz, J., Vladarean, M.-L., McAuley, J., Catasta, M.: Bartering books to beers: a recommender system for exchange platforms. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM '17, pp. 505–514. ACM, New York (2017)
29. Roberts, M., Ducheneaut, N., Begole, B., Partridge, K., Price, B., Bellotti, V., Walendowski, A., Rasmussen, P.: Scalable architecture for context-aware activity-detecting mobile recommendation systems. In: 2008 International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2008. WoWMoM 2008, pp. 1–6. IEEE, Piscataway (2008)
30. Sample—superstore sales. https://community.tableau.com/docs/DOC-1236. Last modified by: Micheal Martin
31. Sanchez, F., Barrilero, M., Uribe, S., Alvarez, F., Tena, A., Menendez, J.M.: Social and content hybrid image recommender system for mobile social networks. Mob. Netw. Appl. **17**(6), 782–795 (2012)
32. Sassi, I.B., Mellouli, S., Yahia, S.B.: Context-aware recommender systems in mobile environment: on the road of future research. Inf. Syst. **72**, 27–61 (2017)
33. Satyanarayanan, M.: Mobile computing: the next decade. In: Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, p. 5. ACM, New York (2010)
34. Stuedi, P., Mohomed, I., Terry, D.: WhereStore: location-based data storage for mobile devices interacting with the cloud. In: Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, p. 1. ACM, New York (2010)
35. Su, J.-H., Wang, B.-W., Hsiao, C.-Y., Tseng, V.S.: Personalized rough-set-based recommendation by integrating multiple contents and collaborative information. Inf. Sci. **180**(1), 113–131 (2010)
36. Szczerbak, M., Toutain, F., Bouabdallah, A., Bonnin, J.-M.: Collaborative context experience in a phonebook. In: 2012 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 1275–1281. IEEE, Piscataway (2012)
37. Taleb, T., Dutta, S., Ksentini, A., Iqbal, M., Flinck, H.: Mobile edge computing potential in making cities smarter. IEEE Commun. Mag. **55**(3), 38–43 (2017)

38. Tarus, J.K., Niu, Z., Kalui, D.: A hybrid recommender system for e-learning based on context awareness and sequential pattern mining. Soft. Comput. 1–13 (2017)
39. Turban, E., Whiteside, J., King, D., Outland, J.: Mobile Commerce and the Internet of Things, pp. 167–199. Springer, Berlin (2017)
40. Wang, T., Liu, L.: Privacy-aware mobile services over road networks. Proc. VLDB Endowment **2**(1), 1042–1053 (2009)
41. Wang, S.-L., Chen, Y.L., Kuo, A.M.-H., Chen, H.-M., Shiu, Y.S.: Design and evaluation of a cloud-based mobile health information recommendation system on wireless sensor networks. Comput. Electr. Eng. **49**, 221–235 (2016)
42. Wiesner, M., Pfeifer, D.: Health recommender systems: concepts, requirements, technical basics and challenges. Int. J. Environ. Res. Public Health **11**(3), 2580–2607 (2014)
43. Yang, W.-S., Cheng, H.-C., Dia, J.-B.: A location-aware recommender system for mobile shopping environments. Expert Syst. Appl. **34**, 437–445 (2008)
44. Yang, B., Lei, Y., Liu, J., Li, W.: Social collaborative filtering by trust. IEEE Trans. Pattern Anal. Mach. Intell. **39**(8), 1633–1647 (2017)

Chapter 13
Association Rule Mining in Higher Education: A Case Study of Computer Science Students

Njoud Alangari and Raad Alturki

13.1 Introduction

The volumes of data that the world is producing every day are extremely huge. Data can be produced by humans directly through our interactions with social networks or by feeding of data into electronic systems. Also, they can be produced solely by machines, such as those logging the activities of humans or machines. Data production and storage have become the norm in every aspect of our lives. They are used in healthcare, banking, education, and even in homes. With adaptation of new technologies such as the internet of things (IoT) and advancements in technologies to produce and store data, it is expected that the world will produce more and more data.

Data can be stored in many forms and in different ways. They can be stored in relational databases and spreadsheets, and are called structured data. Alternatively, data can be stored in a nontraditional row–column database, such as text and multimedia content (e.g., videos, photographs, and audio) and are called unstructured data. Furthermore, data that contain semantic tags (such as e-mail messages, XML, and HTML) and are not stored in relational databases are called semistructured data. Because of this diversity, there have been many methods and techniques used to mine each type. Data differ in the scope and the field that they belong to, and data-mining (DM) experts deal with them differently. For instance, the objective used to mine educational data can be different from the objective used to mine healthcare data. It could be acceptable to build a model that predicts student performance with

N. Alangari · R. Alturki (✉)
Department of Computer Science, Al-Imam Mohammad Ibn Saudi Islamic University, Riyadh, Saudi Arabia
e-mail: naalanqari@sm.imamu.edu.sa; ralturki@imamu.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*, EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_13

85% accuracy, whereas it is not acceptable to have accuracy of only 85% to predict the success rate of drug treatment.

DM techniques can be predictive or descriptive. Predictive methods use variables to predict unknown or future values of one or more variables. Examples of predictive methods are classification, regression, and deviation detection [1]. Descriptive methods such as clustering, association rule discovery, and sequential pattern discovery find human-interpretable patterns that describe data [1].

Regression techniques aim to predict the value of a continuous attribute on the basis of the values of other attributes [1]. The simplest form of regression is linear regression, where the class is a linear combination of attributes with predetermined weights. On the other hand, classification methods aim to predict the value of a discrete attribute on the basis of the values of other attributes. Classification has many approaches, including decision tree (DT), Bayesian classifier or network, neural network (NN), support vector machine (SVM), and logistic regression approaches. Logistic regression allows us to use regression for classification.

Clustering measures the similarity between data points or variables; in other words, it groups similar data points on the basis of their attributes, and each group is called a cluster [1]. Clustering methods are divided, in general, into two groups: partitioning methods and hierarchical methods. Association rules produce dependency rules based on strongly associated different attribute values [2]. There are many association rule–mining algorithms such as Apriori, Equivalence Class Transformation (Eclat), and frequent pattern growth (FPGrowth) [3].

Many researchers have used DM techniques such as regression, classification, clustering, and association rule mining in the field of education. They have applied these analyses in traditional education, web-based education (e-learning), and learning management systems [4]. Their studies have been conducted to accomplish many tasks related to students' learning processes, including providing feedback, predicting student performance, and detecting student behavior [4]. In traditional education, there are hidden patterns in data that are difficult for instructors and administrators to notice without the help of DM techniques. Such knowledge could be useful to students' learning processes in many ways. Processes such as students registering in courses, and instructors making or changing major plans or advising students, take time and effort during every semester. With the use of students' historical data that universities collect over the years, DM can help to enhance these processes. As a result, instructors and students can make better-informed decisions.

In this chapter, we report our work in finding an association between courses in a Computer Science (CS) program on the basis of students' grades. We report our work in discovering interesting rules by using different parameters such as support, confidence, lift, Kulczynski (Kulc), and the imbalance ratio (IR). We used the Apriori algorithm to mine data on undergraduate students majoring in CS at our university. The rest of this chapter is structured as follows: we review research in DM and association mining in Sect. 13.2. In Sect. 13.3 we give a detailed description of our experiments in using association rule mining to find interesting rules. We give our conclusion in Sect. 13.4.

13.2 Related Work

In this section, we review some related work that has used DM in education. We start by discussing previous work in terms of the classification methods used and the challenges associated with them. Then, we give a brief background of association rule mining and how we measure rules' interestingness. After that, we review related work that has used association rule mining in education.

13.2.1 Classification in Education

Classification is one of the most popular DM techniques that have been used in research to analyze educational data. Several studies have been conducted to review the literature in this area (e.g., a study by Shahiri et al. [5]) and to compare classification methods that have been used [5, 6]. For instance, Hämäläinen and Vinni [6] compared classification methods used on educational data on the basis of eight general criteria.

Many attributes have been used to predict students' performance, such as previous courses, standardized examination scores, or preuniversity scores. Different studies have used different attributes in their analyses. Shahiri et al. [5] reviewed important attributes used in prediction, such as the Cumulative Grade Point Average (CGPA) [7, 8], internal assessment [8, 9], students' demographic data [10, 11], external assessment, and psychometric factors. According to Shahiri et al. [5], the most popular task is classification, for which many algorithms have been used, such as DT [7–10, 12], artificial neural networks (ANNs) [7, 8, 12], naïve Bayes [9, 10, 12], K-nearest neighbor [10, 11], and SVM. Shahiri et al. [5] mentioned that most researchers have used CGPA and internal assessment as data attributes, and ANN and DT as classification techniques.

Several studies [7–11, 13] have focused on predicting students' performance by using DM techniques alone. The prediction of overall performance or performance in a specific course could be based on different attributes such as students' preuniversity data [7, 10], average of course attendance [11], grades in courses [13], marks in the course [8, 9], and behavioral features in an e-learning education system [12]. Most of the related studies that we reviewed used classification algorithms for prediction, including a nearest neighbor algorithm (IBk) [10], rule learners (OneR and JRip) [9, 10], classification based on an association rules algorithm [13], linear regression models [7], a classification and regression tree (C&RT), and chi-squared automatic interaction detection (CHAID) [8]. To compare the models, the studies used different evaluation measures such as accuracy (which was used by most of the studies), precision, recall, and F-measures [12].

13.2.2 Background to Association Rule Mining

Association rule mining is a DM technique that discovers interesting relations between attributes and then generates rules that represent these relations. These rules do not imply a causal relationship; for example, rule ($A \Rightarrow B$) does not mean that A causes B. However, the rules imply an association relationship between attributes (that is, those attributes go together) [14]. As we have mentioned above, the popular algorithms in association rule mining are Apriori, Eclat, and FPGrowth [3].

The Eclat algorithm uses a vertical data format method where each item is stored together with its list of TIDs (that is, the IDs of transactions that contain this item) [3, 15]. It is used only at the Apriori join step to generate the candidate itemsets [15]. To compute the support for an itemset, Eclat uses an intersection-based approach [15]. The intersection operation on TIDs is fast frequency counting, and it is advantageous for the vertical format that Eclat uses [16]. However, it has a drawback when the intermediate results of vertical TID lists become too large for the memory [16].

The FPGrowth algorithm was developed on the basis of a new data structure called a frequent pattern tree (FP-tree) [17]. It stores the database in the main memory using a combination of vertical and horizontal database layouts [17]; the transactions are stored in a tree structure, and each item has a linked list going through to all transactions that contain that item [17]. It avoids the cost of the candidate generation that Apriori does, by focusing on frequent pattern growth [3]. However, with long pattern data sets, it consumes more memory and performs badly [17, 18].

Apriori is a basic algorithm to find frequent item sets [3]; it is the one that we used in this work. Sometimes, Apriori produces a huge number of rules, making it difficult to use all of them. Because of that, and for other reasons that will be mentioned later, we must use interestingness measures. The measures of interestingness evaluate how interesting each rule is. This makes it easier to distinguish between the rules.

Measures of Interestingness In Apriori, patterns can be represented in the form of association rules. If the rule is ($A \Rightarrow B$), then, to measure its interestingness, we have many measures such as support, confidence, lift, Kulczynski, and the IR. Such measures are useful to distinguish between rules especially when there is a huge number of resulting rules. In addition, we should distinguish between uninteresting rules, which present an *obvious fact*, and new rules that could be interesting and useful. It is common in association rule mining to get a large number of rules that present facts we already know [19]. More details about the measures are as follows:

- The *absolute support* (also known as support count, count, or occurrence frequency) for itemsets A and B is the number of transactions that contain the itemset, and this is the probability: $P(A \cup B)$ [3]. The support for the rule ($A \Rightarrow B$) (sometimes referred to as relative support) is the count of transactions containing A and B to the number of transactions in the database [20].

$$\text{Support}\,(A \Rightarrow B) = \frac{\text{count}\,(A \cup B)}{n} \tag{13.1}$$

- The *confidence* of a rule $(A \Rightarrow B)$ is the number (or count) of transactions that contain A and B to the number of transactions that contain A, and this is the conditional probability: $P(B|A)$ [3, 20]. The confidence value is calculated as shown below [20]:

$$\text{Confidence}\,(A \Rightarrow B) = \frac{\text{count}\,(A \cup B)}{\text{count}(A)} \tag{13.2}$$

- The *lift* of a rule measures how many more times A and B occur together in transactions than would be expected if A and B were statistically independent (not correlated). The lift value could be equal to 1, which would mean that A and B are independent and there is no correlation between them; or it could be less than 1, which would mean that the occurrence of A is negatively correlated with the occurrence of B; or it could be greater than 1, which would mean that A and B are positively correlated, and the occurrence of one implies the occurrence of the other. For example, if the lift of rule $(A \Rightarrow B)$ is greater than 1, then we could say that A occurrence increases (or "lifts") the likelihood of B occurrence by a factor of the lift's value [3]. The lift of a rule can be calculated as follows [20]:

$$\text{Lift}\,(A, B) = \frac{P\,(A \cup B)}{P(A)P(B)}, \text{OR} \tag{13.3}$$

$$\text{lift}\,(A \Rightarrow B) = \frac{\text{confidence}\,(A \Rightarrow B)}{\text{support}(B)} \tag{13.4}$$

- The *Kulczynski* measure of A and B (abbreviated as Kulc) is an average of two confidence measures where the two confidence measures mean the conditional probabilities: the probability of itemset B given itemset A, and the probability of itemset A given itemset B. Its range is from 0 to 1, and the higher the values are, the closer the relationship between A and B is [3]. Kulc = 0.5 signifies neutral or balanced skewness, whereas the further the value is from 0.5, the closer the relationship is between the two item sets [21]. Kulc is defined by Eq. (13.3).

$$\text{Kulc}\,(A, B) = \frac{1}{2}\,(P\,(A|B) + P\,(B|A)) \tag{13.5}$$

- The *imbalance ratio* (IR) assesses the imbalance of two item sets (A and B) in rule implications [3]. Its range is from 0 to 1; IR = 0 means that the two directional implications between A and B ($A \Rightarrow B$ and $B \Rightarrow A$) are the same, which means it is not an interesting rule, whereas IR = 1 means it is a highly skewed or very interesting rule [21]. IR is calculated by Eq. (13.3):

$$\mathrm{IR}(A, B) = \frac{|\sup(A) - \sup(B)|}{\sup(A) + \sup(B) - \sup(A \cup B)} \tag{13.6}$$

Mining of association rules is a two-step process: first, we must find all frequent item sets that satisfy the minimum support threshold (Min_sup) specified by the user. Second, from these item sets, we must generate association rules that satisfy the minimum confidence threshold (Min_conf), and these rules are called strong [3]. However, when the items that we are interested in have support that is below (or far below) a user-specified minimum support threshold, they are called infrequent (or rare) items [22]. Rare items are caused by an imbalance in the data set where some items have a very high frequency, count, or support, while other items have very low support, and the resulting rules mostly cover only those items with high support [3, 22]. To mine the rare items, there are several methods that can be applied, such as balancing techniques or rare association rule–mining algorithms [23]. Another way is to set the minimum support threshold to a low value, which counts as the simplest way to mine rare items [22].

If we choose to set the minimum support threshold at a low value, that will produce a huge number of strong rules. When the resulting strong rules are huge in number, then use of a lift will help to rank or filter these rules [20]. Another reason to use a lift is to avoid misleading "strong" rules, because not all strong rules are interesting [3]. However, the lift is influenced by a null transaction—a transaction that does not contain any of the itemsets being examined. For example, a transaction that does not contain item sets A and B is a null transaction. If the value of a measure (of the interestingness of a rule) is not influenced by null transactions, it is called a *null-invariant measure*.

Null invariance is an important property for measuring association patterns in large transaction databases. So, a lift is not a null-invariant measure, whereas Kulc and IR are null-invariant measures because they are not influenced by a null transaction [3, 21]. Because of that, the lift has difficulty distinguishing interesting pattern association relationships in comparison with Kulc and IR. As far as we know (from reviewing the studies by Jiawei [3], Gupta and Arora [21], Wu et al. [24], and Gopalakrishnan [25]), we could use the three measures lift, Kulc and IR together as follows:

- If the Kulc value is close to 1, then the left-hand side (LHS) and the right-hand side (RHS) are positively correlated [3, 24].
- If the Kulc value is close to 0, then the LHS and RHS are negatively correlated [3, 24].
- If Kluc = 0.5 (that is, neutral) [3, 21, 24], then check the IR value.
  - If IR = 0, then it is not an interesting rule [21].
  - If the IR value is close to 1, then the rule might be worth looking at [21].

13.2.3 Association Rule Mining in Education

Association rule mining has been applied in education, mostly using the Apriori algorithm. The relevant studies have used Apriori for several objectives. Some of them have used it to predict students' performance [14, 26, 27] and to provide a good placement for a student by matching the organization's requirement with the student's profile [26]. In a study by Kasih et al. [14], the prediction of the students' final results was based on their performance in eight courses in the first four semesters, while in a study by Borkar and Rajeswari [27], the prediction was achieved by finding the association between attributes such as attendance and assignments. In a study by Ahmed et al. [28], the authors used students' academic and personal data to discover their impact on the students' performance; they extracted the association rules related to the impacts of sex, residence, retention, etc.

Other studies have used students' admission data [29, 30]. In a study by Mashat et al. [29], the data represented applicant student information and their status of being rejected or accepted for enrollment at the university. The researchers applied Apriori to the whole data set and then to the accepted applicants and the rejected applicants separately. The resulting rules were presented and interpreted with respect to the admissions office perspective [29]. Abdullah et al. [30] applied the SLPGrowth (Significant Least Pattern Growth) algorithm and two measures—lift and critical relative support (CRS)—to a student admission data set.

Damaševičius [31] aimed to improve the content of an informatics course. He used association rules and ranked course topics on the basis of their importance to the final course marks. He also proposed a novel metric called "cumulative interestingness" for assessing the strength of an association rule. Vranic et al. [19] used data on an electrical engineering fundamentals course with general data to predict the success of the next year's students in this course.

Upendran et al. [32] proposed a course recommendation system that suggested courses for new students. They used the Apriori algorithm to generate rules using previous students' marks in core courses and focused on rules with success as a consequence. These rules were used to suggest courses for new students where they had a high probability of success.

Some studies have associated courses with students' grades; for example, Buldu and Üçgün [33] were interested in the relation between courses in which students failed. In the study by Ahmed et al. [28], the resulting association rules showed that the grade in one course might depend on prerequisite courses. In the study by Upendran et al. [32], marks in core subjects such as Mathematics, Physics, Chemistry, Biology, Computer Science, and English were considered as attributes. Table 13.1 summarizes some of the studies' experimental settings and the Apriori parameters that they applied.

Table 13.1 Summary of studies that have used the Apriori algorithm

Study	Number of students/records	Min_supp	Confidence	Other parameters	Number of rules
Angeline [26]	21 students	0.33	0.75	Max rule length = 4, lift filtering = 1.1	127
Borkar and Rajeswari [27]	60 students	0.01	0.9, 0.87, and 0.7		2, 2, and 11
Kasih et al. [14]	146 records	0.2	0.8	–	6
Mashat et al. [29]	83K records (accepted and rejected applications)	0.4	0.75		4
	38K records (accepted applications)	0.6	Not mentioned	–	5
	45K records (rejected applications)	0.6	Not mentioned	–	3
Vranic et al. [19]	952 records	0.4	0.8	–	>500

Min_supp minimum support threshold

13.3 Experimental Settings and Results

In this section, we present some of the experiments that we did to mine association rules by using the Apriori algorithm to find associations between CS courses based on the students' grades. We present the settings of the experiments, including the data preprocessing, and the results of these experiments.

13.3.1 Experimental Settings

In this subsection, we show how we preprocessed the data, then we show how we selected the values of the Apriori parameters.

Data Preprocessing Our aim in this study was to find an association between CS courses based on students' grades. Therefore, the items in the data set should be in the form (course = G) where course is the course code and G is the grade, G ∈ {A, B, C, D, F}. For example, CS140 = A, CS140 = C, CS322 = A, and CS322 = F.

We started by translating the raw data set from Arabic into the English language, and we selected CS students' data only. Then we transformed the "date of birth" attribute from "dd/mm/yyyy" to "yyyy"; the results are presented as a screenshot (Fig. 13.1). Notice that a student is represented by multiple rows. For instance, if a student studied 50 courses, then he or she would have 50 rows: one row for each course.

ID	Gender	DOB-year	Academic Status	Number of postponing	Starting Major	Cumulative GPA	Semester GPA	Grade of course	Course Number
55989	M	1983	Graduate	1	CS	2.01	2	C+	DAW124
70706	F	1986	Graduate	2	CS	3.04	0	NP	MATH011
70706	F	1986	Graduate	2	CS	3.04	0	NP	MATH012
67365	M	1985	Withdrawn - From U	3	CS	2.72	2.53	F	CS202
71628	M	1987	Graduate	0	CS	3.13	0	NP	MATH011

Two rows for the same student

Fig. 13.1 Data set after deletion of some attributes

ID	Gender	DOB	Number of postponing	Starting Major	ACC100	ARB104	BUS100	COM207	CS104	CS106	CS140
101171	F	1990	0	CS	A	A	A	A	A	A	A
101267	M	1990	0	CS	A	A	B	B	B	C	A
101389	M	1990	0	CS	A	A	B	A	A	A	B
101591	M	1990	0	CS	A	A	C	A	D	D	D
101846	M	1990	0	CS	A	B	A	B	C	B	D
101875	M	1990	0	CS	B	B	A	B	B	B	D

One row for each student

Fig. 13.2 Data set after transformation of rows to columns

After that, the grades were merged into five categories (A, B, C, D, F): A+ was merged with A, B+ was merged with B, etc. Any grades that did not belong to any of the five categories were deleted. For instance, the grades in course CS480 (Practical Training) had two values—NP signified success without a degree and NF signified failure without a degree—and most of the students got NP, so we deleted them. We also deleted the attributes that did not seem useful, especially when they had the same value for all students, such as major and residential area attributes.

Then, to represent each student by one row, we transformed rows into columns; the results are presented as a screenshot (Fig. 13.2), where we present some of the attributes. With this transformation, we kept only the first trail (which is the first grade that a student got when he or she studied the course for the first time). To handle missing data, we selected the graduated students, since their records had fewer missing data/grades. We had 833 CS graduated students.

At the end, to prepare the data set for association rule mining, and to find the association between courses, we selected only the attributes that represented CS courses—general courses and computer specialized (mandatory and elective) courses—which meant that we deleted the rest of the attributes. In the CS study plan, we had 60 courses; for each course, student could get one of five grades (A, B, C, D, F), which meant we had ($60 \times 5 = 300$ items) in the form (course = G).

Setting of Apriori Parameters

After conducting many experiments and reading related works, we set the minimum support at 0.01 and confidence at 0.8. This minimum support meant that approximately 94% of the items would be included in the mining process, while items that had support of less than 0.01 would not be included. It also meant 1% of students, which was either 8 students out of 833 students ($1\% \times 833 = 8.33$) or 4 students out of 483 students [29].

Table 13.2 Kulczynski (*Kulc*) and imbalance ratio (*IR*) values and their meanings

Relationship between LHS and RHS	Kulc/IR	Value
Positive correlation	Kulc close to 1	Kulc $\geq$ 0.7
Negative correlation	Kulc close to 0	Kulc $\leq$ 0.3
Neutral	Kulc close to 0.5	0.7 > Kulc > 0.3
Very imbalanced	IR close to 1	IR $\geq$ 0.8
Imbalanced	IR relatively close to 1	IR $\geq$ 0.6
Balanced	IR close to 0	IR $\leq$ 0.3
Neutral	Kulc and IR close to 0.5	0.6 < IR < 0.3 and 0.7 > kulc > 0.3

LHS left-hand side, *RHS* right-hand side

Besides using support and confidence to measure the rules' interestingness, we decided to use lift, Kulc, and IR for the reasons mentioned in Sect. 13.2. In addition, on the basis of the relation between the three measures, we wrote the pseudocode below to evaluate the rules. As in the studies by Bramer [20] and Angeline [26], we used lift to rank or filter the rules, then we applied the pseudocode. Table 13.2 lists the values that we chose for Kulc and IR.

The itemset number was set at (2, 3, 4, and 5) to specify the length of the rules. This helped us to focus on each subset of the rules; for example, when we analyzed 2-itemset rules, we focused on the relationship between two courses only, so if the relation was positive, then getting a high grade in one course would be associated with getting a high grade in the other one, and vice versa.

```
Pseudocode Used for Rule Filtering
  Sort according to lift and lift > 1
  If Kulc close to 1 then
       LHS and RHS are positively correlated
  Else if Kulc close to 0 then
       LHS and RHS are negatively correlated
  Else if Kulc close to 0.5 then
       // Neutral, use IR to help find the imbalance
       If IR very close to 1 then
       // Imbalanced
         The rule might be worth looking at: very imbalanced case
       Else if IR relatively close to 1 then
       // Imbalanced
            The rule might be worth looking at: imbalanced case
       Else if IR close to 0 then
       // Balanced
            Not interesting rule
       Else "neutral"
       // Kulc close to 0.5 and IR between 0.3 and 0.6
```

13.3.2 Results of the Experiments

In the experiments, we used two data sets. In the first data set, the instances were CS graduated students (833 students). In the second data set, we kept students who failed at least one course (483 students or rows) and we kept items that contained failing grades only (CS140 = F, CS322 = F, etc.). We present the two experiments as follows:

- The first experiment focused on the association of courses based on success and failure, and it used data set 1.
- The second experiment focused on association of courses based on failure, and it used data set 2.

Also, for each experiment, we generated rules with itemset numbers from 2 to 5. Because of space limitations, we present and discuss the resulting rules only for 2- and 3-itemsets, presenting 2-itemset rules for the first experiment and 3-itemset rules for the second experiment.

Association of Courses Based on Success and Failure In this experiment, we set Apriori parameters as listed in Table 13.3.

The results of the 2-itemset were 1138 rules, and we were interested in rules where the LHS and RHS were positively correlated (103 rules) or worth looking at, especially in a very imbalanced case (453 rules). We first explore and present the positive correlation rules, then those worth looking at. After that, we present the rules that contained grade F.

Positive Correlation Rules In Table 13.4, we present the top five rules with the highest lift values where LHS and RHS were positively correlated. As can be seen, these rules had high confidence and the support values were relatively low to high, being in the range of 0.3–0.8, which meant that 30–80% of the students achieved these grades in these courses. The confidence values were in the range of 0.8–0.9, indicating that these rules were found to be true 80–90% of the time; in other words, in 80–90% of instances where a student achieved LHS, he or she would achieve RHS too.

As an example of the rules presented in Table 13.4, **rule (1043)** {**ARB104=A**} ⇒ {**IDE133=A**} **(sup = 0.384 and conf = 0.821)** means that 38% of the students achieved an A in both course ARB104 and course IDE133, and the rule was found

Table 13.3 Apriori parameter values

Parameter	Value
Min_supp	0.01
Confidence	0.8
Itemset number	2, 3, 4, and 5

Min_supp minimum support threshold

Table 13.4 Top five 2-itemset rules where the left-hand side (*LHS*) and right-hand side (*RHS*) are positively correlated

Rule no.	Rules	Support	Confidence	Lift	Kulc	IR	LHS and RHS correlation
1043	{ARB104 = A} ⇒ {IDE133 = A}	0.384	0.821	1.312	0.717	0.222	Positive
1053	{ECO100 = A} ⇒ {QUR401 = A}	0.415	0.824	1.241	0.725	0.212	Positive
1062	{BUS100 = A} ⇒ {COM207 = A}	0.403	0.806	1.188	0.700	0.229	Positive
1063	{BUS100 = A} ⇒ {QUR451 = A}	0.419	0.837	1.186	0.715	0.261	Positive
1055	{ECO100 = A} ⇒ {QUR451 = A}	0.421	0.836	1.184	0.716	0.256	Positive

IR imbalance ratio, *Kulc* Kulczynski

Table 13.5 Number of 2-timeset rules containing one of the Holy Quran (*QUR*) courses

QUR on RHS	Number of rules
{QUR101 = A}	139
{QUR201 = A}	127
{QUR251 = A}	105
{QUR351 = A}	103
{QUR301 = A}	89
{QUR151 = A}	79
{QUR451 = A}	75
{QUR401 = A}	41

RHS right-hand side

to be true for 82% of those students; in other words, in 82% of instances where a student got an A in ARB104, he or she would get an A in IDE133 too.

In addition, we noted the following about the positive correlation rules:

- All rules that had a positive correlation associated getting grade A in LHS courses with getting grade A in RHS courses.
- Examples of the courses that appeared in the rules are general courses and two mandatory courses, CS492 and CS493 (Senior Project in Computer Science 1 and Senior Project in Computer Science 2). However, grade A was the only grade that appeared.
- Most of these rules were in the form ({X = A} ⇒ {QURxxx = A}), where "X" represented a course and "xxx" represented the code of the QUR (Holy Quran) courses. Table 13.5 shows the number of rules (in this form) for each QUR course.

The high support indicates that these item sets {course1 = A, course2 = A} appeared frequently in the data set; most of the students got a grade of A in these courses (the general courses and two mandatory courses (CS492 and CS493)). For example, Fig. 13.3 shows the frequency or support of the QUR courses with grades

(A, B, C, D, F) and, as we said, (QURxxx = A) items were the most frequent. As a conclusion, these rules may not be interesting, since getting an A in these courses could count as an obvious fact, and any rule that supports an obvious fact is not an interesting rule.

Rules Worth Looking At In Table 13.6 we present the top five rules that were worth looking at (very imbalanced cases). These rules had high confidence values and low to relatively low support values; the support values were in the range of 0.01–0.1, meaning that 1–10% of the students achieved grades (A, B, C, D, F) in the LHS courses and grade A in the RHS courses. The confidence values are in the range of 0.8–1, which indicates that these rules were found to be true 80–100% of the time; in other words, 80–90% of instances where a student achieved in LHS courses, he

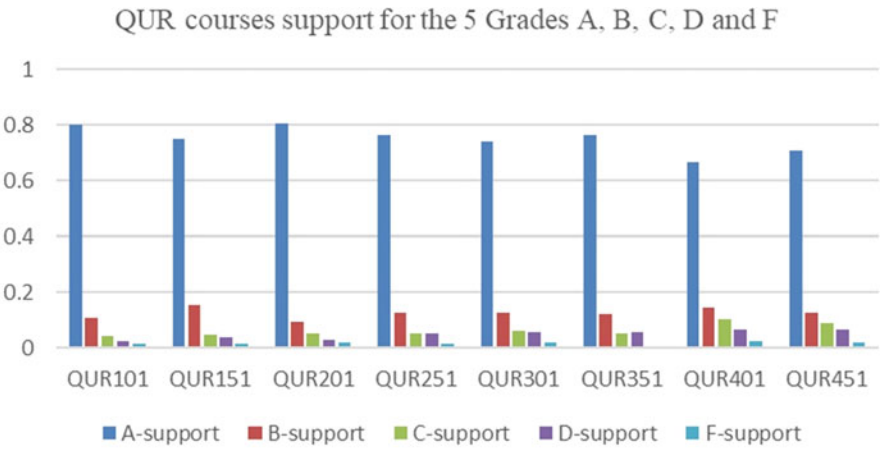


Fig. 13.3 Support of Holy Quran (*QUR*) courses with grades (A, B, C, D, and F)

Table 13.6 Top five "worth looking at" 2-itemset rules

Rule no.	Rules	Support	Confidence	Lift	Kulc	IR	LHS and RHS correlation
45	{CS439 = A} ⇒ {CS322 = A}	0.014	0.800	4.660	0.442	0.877	Very imbalanced case
46	{CS439 = A} ⇒ {CS370 = A}	0.014	0.800	4.191	0.438	0.889	Very imbalanced case
47	{CS439 = A} ⇒ {MATH227 = A}	0.014	0.800	3.471	0.431	0.908	Very imbalanced case
106	{CS438 = A} ⇒ {CS391 = A}	0.025	0.955	2.432	0.509	0.930	Very imbalanced case
49	{CS439 = A} ⇒ {ENG208 = A}	0.018	1.000	2.394	0.522	0.957	Very imbalanced case

IR imbalance ratio, *Kulc* Kulczynski, *LHS* left-hand side, *RHS* right-hand side

or she would achieve in RHS courses too. A "worth looking at" rule may or may not imply an interesting relationship.

We noticed the following about the "worth looking at" rules:

- All rules that had a positive correlation associated getting grade (A, B, C, D, F) in LHS courses with getting grade A in RHS courses.
- As we saw in the (2-timeset rules with positive correlation), most of the rules were in the form (X = G ⇒ {QURxxx = A}), where G represented one of the grades (A, B, C, D, F), and we said that these rules were mostly not interesting, since they could count as obvious facts because of the high support of (QURxxx = A) items.

As an example of the rules presented in Table 13.6, **rule (45)** {**CS439 = A**} ⇒ {**CS322 = A**} implied that maybe there was an association between getting an A in both course CS439 (Cloud Computing) and course CS322 (Operating Systems), and we know from the CS study plan that one of the prerequisite courses to register in CS439 (Cloud Computing) was success in CS322 (Operating Systems). So, an association between getting an A in both courses is logical, since it confirms what we already know. The same applies to **rule (46)** {**CS439 = A**} ⇒ {**CS370 = A**}, where there might an association between getting an A in both course CS439 (Cloud Computing) and course CS370 (Introduction to Databases), and since the CS study plan did not link these two courses, that could mean that this rule is a new rule and therefore could be an *interesting rule*.

Table 13.7 shows some of the rules that contained CS courses on the LHS, arranged by the support values from largest to smallest. For example, the first three rules applied to approximately 5% of the students (5% of 833 students = 41 students), and they were true 90% or 80% of the time. They showed the association between getting an A in CS401 (Computational Numerical Analysis) and getting an A in CS493 (Senior Project in Computer Science 2), as in **rule (262)**; getting

Table 13.7 "Worth looking at" 2-itemset rules where the left-hand side (*LHS*) items are Computer Science courses

Rule no.	Rules	Support	Confidence	Lift	Kulc	IR	LHS and RHS correlation
262	{CS401 = A} ⇒ {CS493 = A}	0.059	0.907	2.026	0.519	0.844	Very imbalanced case
258	{CS401 = A} ⇒ {CS391 = A}	0.058	0.889	2.264	0.518	0.820	Very imbalanced case
264	{CS401 = A} ⇒ {CS492 = A}	0.058	0.889	1.903	0.506	0.848	Very imbalanced case
235	{CS471 = A} ⇒ {CS492 = A}	0.052	0.878	1.879	0.494	0.861	Very imbalanced case
201	{CS451 = A} ⇒ {CS492 = A}	0.036	0.857	1.835	0.467	0.898	Very imbalanced case

IR imbalance ratio, *Kulc* Kulczynski, *RHS* right-hand side

an A in CS391 (Seminar), as in **rule (258)**; and getting an A in CS392 (Senior Project in Computer Science 1), as in **rule (264)**.

"Worth Looking At" Rules Containing Failed Courses We present some of the 2-timeset rules that contain grade F in Table 13.8; they are worth looking at (a very imbalanced case) on the basis of both their Kulc value and their IR value. Note that they have high confidence in the range of 0.8–1 and low support in the range of 0.011–0.067; that is, 1.1–6.7% of the students who got grade F in LHS courses and got grade A in QUR courses (RHS). Also, these rules were found to be true 80–100% of the time. As we know, a "worth looking at" rule may or may not imply an interesting relationship. From our point of view, since most of the students got an A in QUR courses, those students would get different grades (A, B, C, D, F) in the rest of their courses. So, it is an obvious fact, which means these rules do not seem to be interesting. For example, the support value for **rule (13)** is equal to the support value for item (CS430 = F) on the LHS; the same students who got an F in CS430 (approximately 11 students) got an A in QUR courses.

Association of Courses Based on Failure In this experiment, to find the association between failure and getting an F grade in courses, we used the second data set, which contained CS graduated students who failed in at least one course (483 rows), with 60 courses, and each course where a student got an F grade (so, 60 items in form (course = F)). Table 13.9 presents the Apriori parameter values that we used. We present the 3-itemset rules because there was no resulting 2-itemset rule.

Table 13.8 2-Itemset rules that contain an F grade on the left-hand side (*LHS*), on the right-hand side (*RHS*), or both

Rule no.	Rules	Support	Confidence	Lift	Kulc	IR	LHS and RHS correlation
13	{CS430 = F} ⇒ {QUR251 = A}	0.014	1.000	1.312	0.509	0.981	Very imbalanced case
11	{CS412 = F} ⇒ {QUR201 = A}	0.011	1.000	1.241	0.507	0.987	Very imbalanced case
12	{CS430 = F} ⇒ {QUR151 = A}	0.013	0.917	1.228	0.467	0.979	Very imbalanced case
20	{CS451 = F} ⇒ {QUR151 = A}	0.014	0.857	1.148	0.438	0.974	Very imbalanced case
21	{CS451 = F} ⇒ {QUR351 = A}	0.014	0.857	1.121	0.438	0.975	Very imbalanced case

IR imbalance ratio, *Kulc* Kulczynski

Table 13.9 Apriori parameter values

Parameter	Value
Min_supp	0.01
Confidence	0.8
Itemset number	2, 3, 4, and 5

Min_supp minimum support threshold

Table 13.10 Top five "worth looking at" 3-itemset rules for the failure data set

Rule no.	Rules	Support	Confidence	Lift	Kulc	IR	LHS and RHS correlation
18	{CS340 = F, CS471 = F} ⇒ {CS401 = F}	0.012	0.857	8.809	0.492	0.833	Very imbalanced case
17	{CS403 = F, CS471 = F} ⇒ {CS401 = F}	0.010	0.833	8.564	0.470	0.854	Very imbalanced case
29	{CS215 = F, CS370 = F} ⇒ {CS401 = F}	0.010	0.833	8.564	0.470	0.854	Very imbalanced case
13	{QUR101 = F, STA111 = F} ⇒ {MATH227 = F}	0.010	1.000	7.318	0.538	0.924	Very imbalanced case
5	{MATH227 = F, QUR101 = F} ⇒ {CS344 = F}	0.010	1.000	7.103	0.537	0.926	Very imbalanced case

IR imbalance ratio, *Kulc* Kulczynski, *LHS* left-hand side, *RHS* right-hand side

The results of the 3-itemset were 43 rules; there was no rule where LHS and RHS were positively correlated. Table 13.10 shows the top five "worth looking at" rules (a very imbalanced case). These rules had high confidence values and low support values. The support values were in the range of 0.01–0.024, which meant that 1–2.4% of the students achieved grade F in these courses, while the confidence values in the range of 0.8–1 indicated that these rules were found to be true 80–100% of the time; in other words, 80–100% of the time that a student achieved LHS, he or she would achieve RHS too. For example: **rule (18)** {**CS340 = F, CS471 = F**} ⇒ {**CS401 = F**} indicates that 1.2% of the students got an F in all three courses of CS340 (Artificial Intelligence), CS471 (Database Management Systems), and CS401 (Computational Numerical Analysis), and 85% of the times that a student failed in CS340 and CS471, he or she would fail in CS401 too.

13.4 Conclusion

In this chapter, we have reviewed some of the literature that has used data-mining (DM) techniques (such as classification and association rule mining) in education for exploring patterns and extracting knowledge. We conducted experiments to extract and mine interesting rules from data on undergraduate Computer Science (CS) students, using the Apriori algorithm, and we presented the settings of these experiments and some of the results. We answered the questions of the research and we used lift, Kulczynski (Kulc), and the imbalance ratio (IR) to measure the interestingness of rules, along with their support and confidence. We explained how

we preprocessed the data and how we set the values for minimum support, minimum confidence, Kulc, and IR. Our results showed correlation between some courses when students obtained A and F grades. Other grades (B, C, and D) did not show correlation between courses in the 2-itemset rules. In addition, they showed that most of the interesting rules had support higher than 1% and confidence higher than 80%. Therefore, we cannot confirm or deny previous opinions that have associated some courses with each other, such as associating success in mathematics with success in programming. We also plan to apply classification algorithms to the data set.

References

1. Tan, P.-N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Pearson Addison Wesley, Boston (2005)
2. Frank, E., Witten, I.H.: Data Mining: Practical Machine Learning Tools and Techniques, vol. 54, no. 2. Morgan Kaufman, San Francisco (2011)
3. Jiawei, H., Kamber, M., Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. Elsevier, New York (2012)
4. Romero, C., Ventura, S.: Educational data mining: a review of the state of the art. IEEE Trans. Syst. Man Cybernet. Part C Appl. Rev. **40**(6), 601–618 (2010)
5. Shahiri, A.M., Husain, W., Rashid, N.A.: A review on predicting student's performance using data mining techniques. Procedia Comput. Sci. **72**, 414–422 (2015)
6. Hämäläinen, W., Vinni, M.: Classifiers for educational data mining. In: Handb. Educ. Data Mining, Data Min. Knowl. Discov. Ser., pp. 57–71 (2010)
7. Ibrahim, Z., Rusli, D.: Predicting students' academic performance: comparing artificial neural network, decision tree and linear regression. In: 21st Annual SAS Malaysia Forum, 5th September 2007
8. Ogor, E. N.: Student academic performance monitoring and evaluation using data mining techniques. In: Electronics, Robotics and Automotive Mechanics Conference (CERMA 2007), pp. 354–359 (2007)
9. Al-Barrak, M.A., Al-Razgan, M.S.: Predicting students' performance through classification: a case study. J. Theor. Appl. Inf. Technol. **75**(2), 167–175 (2015)
10. Verma, K., Singh, A., Verma, P.: A review on predicting student performance using data mining method. Int. J. Curr. Eng. Sci. Res. **3**(1), 127–132 (2016)
11. Harwati, Alfiani, A.P., Wulandari, F.A.: Mapping student's performance based on data mining approach (a case study). Agric. Agric. Sci. Procedia. **3**, 173–177 (2015)
12. Amrieh, E.A., Hamtini, T., Aljarah, I.: Mining educational data to predict student's academic performance using ensemble methods. Int. J. Database Theory Appl. **9**(8), 119–136 (2016)
13. Badr, G., Algobail, A., Almutairi, H., Almutery, M.: Predicting students' performance in university courses: a case study and tool in KSU Mathematics Department. Procedia Comput. Sci. **82**, 80–89 (2016)
14. Kasih, J., Ayub, M., Susanto, S.: Predicting students' final passing results using the Apriori algorithm. World Trans. Eng. Technol. Educ. **11**(4), 517–520 (2013)
15. Pramod, P.S., Vyas, O.P.: Survey on frequent itemset mining algorithms. Int. J. Comput. Appl. **1**(15), 94–100 (2010)
16. Zaki, M.J., Gouda, K.: Fast vertical mining using diffsets. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining—KDD'03, p. 326 (2003)

17. Solanki, S., Soni, N.: A survey on frequent pattern mining methods Apriori, Eclat, FP growth. Int. J. Comput. Tech. **X**(X), (2014)
18. Suman, M., Anuradha, K.T., Ramakrishna, A.: A frequent pattern mining algorithm based on Fp-tree structure and Apriori algorithm. Technicaljournals.Org. **2**(1), 114–116 (2012)
19. Vranic, M., Pintar, D., Skocir, Z.: The use of data mining in education environment. In: 2007 9th International Conference on Telecommunications, pp. 243–250 (2007)
20. Bramer, M.: Principles of Data Mining. Springer, New York (2016)
21. Gupta, D., Arora, H.: Market basket analysis using Apriori and correlation measures. Int. J. Innov. Res. Sci. **6**(6), 10282–10286 (2017)
22. Romero, C., Romero, J.: Mining rare association rules from e-learning data. In: 3rd International Conference on Educational Data Mining, pp. 171–180 (2010)
23. Koh, Y.S., Nathan, R.: Rare association rule mining: an overview. In: Rare Association Rule Mining and Knowledge Discovery: Technologies for Infrequent and Critical Event Detection, IGI Global, pp. 1–14 (2009)
24. Wu, T., Chen, Y., Han, J.: Re-examination of interestingness measures in pattern mining: a unified framework. Data Min. Knowl. Discov. **21**(3), 371–397 (2010)
25. Gopalakrishnan, A.: A Multifaceted Data Mining Approach to Analyzing College Students' Persistence and Graduation. San Francisco State University, San Francisco (2016)
26. Angeline, D.M.D.: Association rule generation for student performance analysis using Apriori algorithm. SIJ Trans. Comput. Sci. Eng. Appl. **1**(1), 12–16 (2013)
27. Borkar, S., Rajeswari, K.: Predicting students academic performance using education data mining. Int. J. Comput. Sci. Mob. Comput. **2**(7), 273–279 (2013)
28. Ahmed, S., Paul, R., Hoque, A.S.M.L.: Knowledge discovery from academic data using Association Rule Mining. In: 2014 17th International Conference on Computer and Information Technology (ICCIT), pp. 314–319 (2014)
29. Mashat, A.F., Fouad, M.M., Yu, P.S., Gharib, T.F.: Discovery of association rules from university admission system data. Int. J. Mod. Educ. Comput. Sci. **5**(4), 1–7 (2013)
30. Abdullah, Z., Herawan, T., Deris, M.M.: Discovering interesting association rules from student admission dataset. Lect. Notes Electr. Eng. **285**, 135–142 (2014)
31. Damaševičius, R.: Analysis of academic results for informatics course improvement using association rule mining. In: Information Systems Development: Towards a Service Provision Society, pp. 357–363 (2009)
32. Upendran, D., Chatterjee, S., Sindhumol, S., Bijlani, K.: Application of predictive analytics in intelligent course recommendation. Procedia Comput. Sci. **93**, 917–923 (2016)
33. Buldu, A., Üçgün, K.: Data mining application on students' data. Procedia. Soc. Behav. Sci. **2**(2), 5251–5259 (2010)

Chapter 14
SelecWeb: A Software Tool for Automatic Selection of Web Frameworks

Thaha Muhammed, Rashid Mehmood, Ehab Abozinadah, and Sanaa Sharaf

14.1 Introduction

The software applications revolution has helped the development of many new distributed and collaborative urban systems [1–16], paving the way for integrated systems, and hence smart cities and societies, see, e.g., [17] for background on smart cities and societies.

Web applications and services are fundamental to designing smart infrastructure and cities. Web frameworks have become an integral part in the development of web applications and services. A software framework is a scaffold structure inside which other applications can be developed. A framework comprises libraries, services, scaffold programs, scaffold codes, interfaces, APIs (application programming interfaces), and other components required for application development. A framework provides the basic building blocks required for the development of application.

A software framework that has been developed particularly for assisting web application development is called a web application framework. It comprises necessary components and services required for the construction of feature rich applications by automating the common web development functions. Most of the web application development frameworks implement the MVC (model–view–

T. Muhammed (✉) · S. Sharaf
Department of Computer Science, FCIT, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: m.thaha.h@ieee.org; ssharaf@kau.edu.sa

R. Mehmood
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: rmehmood@kau.edu.sa

E. Abozinadah
Department of Information Systems, FCIT, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: eabozinadah@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_14

controller) design pattern as shown in Fig. 14.1. It also incorporates various services like versioning using git, svn, or other version management systems and searching. This helps the application in leveraging the framework services for the production of quality applications. Web frameworks also provide user interface elements and powerful ORM (object-relational management).

Web application frameworks promote code reuse and reduce the resource requirements such as time and effort to build and maintain applications. In recent years, a plethora of frameworks have been developed with various features. There is no single all-feature-encompassing framework. Each framework has its own advantages and disadvantages. Suitability of various web frameworks to various application domains varies. Programmers may choose from a variety of web frameworks, and different languages that support them, each with its own strengths and weaknesses. Organizations work in different application domains and have diverse priorities and constraints with regard to the development of applications and services.

In this paper, we propose an automatic tool for selecting a web framework based on a set of criteria and developer preferences. The set of selection criteria is developed by us and is a contribution of this paper. The tool is called SelecWeb. It currently uses analytic hierarchy process (AHP) for comparison, analysis, and decision-making. We provide a detailed description and analysis of the tool including a case study for web framework selection.

The rest of the paper is organized as follows: Sect. 14.2 gives a review of the literature. Section 14.3 introduces the web frameworks that we have been used in this paper. These are Ruby on Rails, Spring, Django, and CodeIgniter. Section 14.4 discusses the selection criteria including the developer and user criteria. Section 14.5 explains the evaluation process and discusses evaluation of each of the selected frameworks. Section 14.6 summarizes the weaknesses and strengths of each

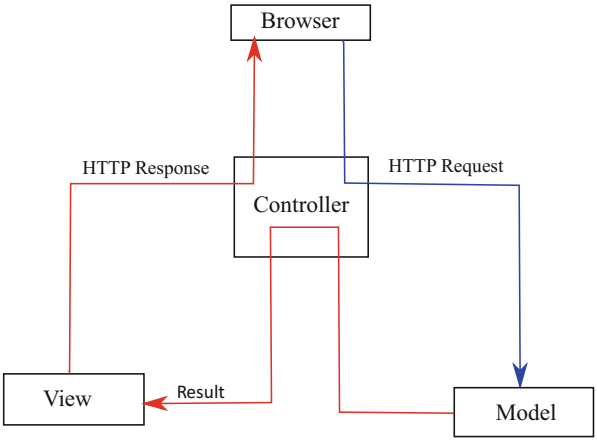


Fig. 14.1 The MVC architecture for web frameworks

framework. Section 14.7 provides a case study that uses AHP for selecting the best framework satisfying requirements of a given application. Section 14.8 concludes the paper along with some design ideas on future extension of the tool using machine learning.

14.2 Literature Survey

Many researchers have evaluated single frameworks in isolation. Numerous works evaluate primary technologies for web applications such as HTML5 [18]. Analysis of web framework application development often evaluates a single web framework, but do not compare it to competing ones. Bachle and Ritscher [19] analyzed and benchmarked Rails. Arthur and Azadegan [20] assessed Spring, a Java-based framework. The researchers evaluate the framework and do not compare it to other competing frameworks. These works mainly list out the features and capabilities of the framework. Matt [21] compares various Java-based frameworks such as Spring, Wicket, Grails, Play, and JRuby. A comparison of Eucalyptus, Apache Hadoop, and the Django–Python stack can also be found [22]. These comparisons deal with frameworks based on the same language. Smutny [23] briefly compares selected web-based mobile frameworks. However, he does not propose a set of criteria for doing so. Henning et al. make a comparison of four mobile web frameworks [24], where they compare HTML5, Sencha Touch, Google Web Toolkit, M-Project, jQtouch, and jQuery. Here, the comparison is based on a set of criteria developed by the researchers. We can reach a conclusion that most of the comparisons lack the backing of scientific criteria.

14.3 Web Frameworks

This section examines the web application framework and introduces four frameworks that will be analyzed in Sect. 14.5.

14.3.1 General

A web framework is a collection of packages and modules that helps web developers to write web applications, web services, and dynamic websites without worrying about low-level details [25]. Frameworks ease the overhead associated with common web development activities. Many frameworks provide libraries for database access, provide support for a number of activities such as interpreting requests, templating frameworks, producing responses, manage sessions, and they promote code reuse often [26].

Many of the major frameworks such as Rails, Django, and Spring are server-side technologies but in the recent years due to advancements in client-side technology such as node.js and CoffeeScript, browsers can be used as a full blown framework stack.

World Wide Web in its infancy used static pages hand coded in HTML which were hosted on web servers. Any alteration to the website required explicit changes from the developers [25]. In earlier days, Common Gateway Interface [25] was used to serve web pages to the web browser. In CGI, we had to program each and every detail of the connection. Each request that arrives at CGI creates a new thread. As the number of requests increases, the number of threads also increases, which might crash the server. New languages for web development like PHP emerged around same time.

Most of these web languages used a spaghetti styled coding where everything starting from HTML views, database access, and other low-level details such as protocols, thread management, and socket management had to be hand coded [26]. These required various libraries that usually did not come with the language and had to be compiled by the developer.

Later on, came frameworks that constitute necessary components and services required for the construction of feature affluent and erudite systems. For instance, Ruby on Rails, Django, Symfony, Zend, CakePHP, CodeIgnitor, Spring, Grails, etc., are some web frameworks.

14.3.2 Ruby on Rails

Rails is a web framework written in Ruby by David Heinemeier Hansson [27]. He derived it from Basecamp, formerly Signal37, a project management tool. Rails increases the productivity of the developer by reducing the line of codes to achieve the end result. It accomplishes more in the least number of lines as compared to other languages such as PHP. Rails is based mainly on two principals: convention over configuration and DRY (don't repeat yourselves). It has an MVC (model–view–controller) architecture. Rails presumes that there is a perfect way to code, and sets these as conventions to be followed while coding in Rails. This results in higher productivity as configuring each and every minute configuration of the application is not required. The second principle called DRY (don't repeat yourselves) states every unique function should only have a single piece of code that accomplishes the task. This results in a more maintainable and less buggy code. Rails is released under the MIT License. It was released in the year 2003. Twitter and Github are the two major websites created with Rails.

14.3.3 Django

Django is a python based free open-source web application framework that was created by Adrian Holovaty and Simon Willison. Currently, it is maintained by an organization called Django Software Foundation. Django has many similarities to Ruby on Rails. Django has an MVC architecture. A cardinal advantage of such a concept is that components are loosely coupled which implies that a database architect's work will not depend on the programmer's or the designer's work. All three can work independently. Amazon.com, craigslist.org, and washingtonpost.com are some of the major applications built with Django. Django is licensed under the BSD License.

14.3.4 Spring

Spring is a web framework which is also called the father of frameworks, due to the fact that it provides scaffold to other Java-based frameworks. Some of the major frameworks that receive support from Spring are Hibernate, EJB, Struts, JSF, etc. [28]. In 2003, Rod Johnson created Spring. Spring is a Java-based framework helpful in creating Java Enterprise applications. Spring combines various components. It is highly valuable when you might want to use different components or various combinations of components in different environments with various configurations. Spring is a framework based on a pattern called dependency injection, which is issued to build highly decoupled systems [28, 29]. Spring consists of an MVC framework, validation framework, and transactional control of databases. It segregates service layer, business layer, and web layer. But what it really does best is injection of objects. In dependency injection [30], the objects are designed such that they receive instances of objects externally from other sources, instead of creating them inside the actual code. This improves decoupling and simplifies testing.

14.3.5 CodeIgniter

CodeIgniter [28] is a web-based framework for building dynamic websites based on PHP [28]. It was created by Rick Ellis in 2006, from ExpressionEngine, a CMS (content management system) owned by Ellislab in 2006. In 2014, the ownership of CodeIgniter was transferred to the British Columbia Institute of Technology, which inculcated it in their core syllabus. CodeIgniter is a lightweight, fast framework with a minimal footprint that helps in rapid application development. PHP's creator Rasmus Lerdorf, an outspoken critic of frameworks, praised CodeIgniter due to its speed and light weightlessness. It's loosely based on MVC architecture. It does not enforce the MVC pattern upon the developers. Controllers are necessary for the

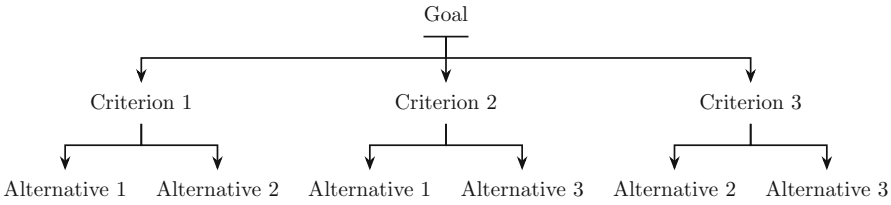


Fig. 14.2 Analytic hierarchy process

development, but models and views are optional, unlike other frameworks such as Rails and Django that enforce MVC strictly. It is a lean MVC framework, with enough capabilities to increase your productivity, at the same time providing third-party modules for extra functionality. The source code of CI is available online at Github [28]. Earlier, CI versions had an Apache-BSD open-source license. Later it was switched to the MIT License.

14.4 Selection Criteria

Web framework selection is an optimization problem [24] because we need to maximize the goals of the organization. We can measure complex requirement criteria by dividing them into sub-criteria and can use functions such as analytic hierarchy process [31] to evaluate final decision criteria as shown in Fig. 14.2. If the decision process is broken down into smaller manageable components, it results in an improved decision-making. Each of these criteria can be given a weight, according to the goals of an organization. Evaluating frameworks based on a single criterion is difficult and a vague measure. Hence, we need to develop a set of criteria, for the purpose of evaluation of frameworks. The overall goal of the decision criteria is to allow an organization to select an optimal framework based on their requirements. Hence, we divide the criteria into two, first from a developers' viewpoint, in terms of usability and second, from the users' viewpoint. A users' viewpoint and their experience with the application is of paramount importance. From developers' viewpoint, we consider the experience and decisions of the developer, such as licensing and cost. Criteria can be classified in two categories: qualitative and binary. Binary criteria are criteria that can be answered using either using yes or no. It examines whether a feature is available in the framework. Qualitative criteria deals with the quality of the framework. Qualitative criteria are extremely useful for decision making. Hence this article will deal with qualitative criteria.

14.4.1 Developer Criteria

From the viewpoint of the developer we consider the following criteria.

License and Cost Various companies have different policies regarding the license of the components and applications they use. Therefore, we need to consider the cost of licensing an application based on the web frameworks. Open-source licenses such as Apache [32] and MIT [33] can be considered as ideal cases, whereas complications can arise from the use of Copyleft licenses [34] such as GNU [35].

Learning Effort Effort and time are required to learn and comprehend a new web framework. This criterion examines the extent up to which the framework follows the general conventions, the intuitiveness of the framework, resemblance to other programming frameworks, and ease of learning. This also considers the effort and time required to master the framework.

Developing Effort The development effort is proportional to the cost of the development. Even though, requirement phase is independent of the framework used, it does influence the implementation. Development effort comprises of time required for the implementation of applications with the framework. Ease of reuse, good tool support, and an IDE with a graphical user interface are good indicators towards development effort.

Long-Term Viability Selection of a framework for development is a significant investment as all the applications developed by the organization will be tied to the framework. Due to rapidly changing technology the web frameworks require frequent updating. A framework with a robust team of developers commercially backed by organizations has greater potential to thrive. Popularity and frequent updates are two other indicators that imply long-term feasibility.

Documentation and Support Good documentation and support increase the speed of mastering the framework. A good quality documentation provides exceptional tutorials and references. Textbooks can be the starting point for popular frameworks. Forums and community provide extra assistance that helps the developer tremendously.

Adaptability Due to the evolution of the technology it may be necessary to modify the framework with extra functionality. This will be easier if the framework provides a module plugin mechanism. This criterion also evaluates the efficacy and availability.

Maintainability The code has to be maintainable. The source code has to be comprehensible and reusable. Modularity and decoupling of the components in framework increase the maintainability. These are the major indicators for maintainability.

14.4.2 User Criteria

From the viewpoint of the user, we can consider the following criteria.

Inherent Look and Feel The acceptance of a web application by a user mainly depends upon the look and feel of the application. Different frameworks provide various themes and feel. A framework should be able to provide a platform-specific theme. We see how the theme provided by the frameworks resembles the native look and feel unless the framework provides means to modify its user interface elements.

Load Time The web application has to be fast and load even on unstable and slow networks. The load time depends upon the complexity of the web framework. The web frameworks can use asynchronous JavaScript or cache the commonly used pages to increase the load speed. This criteria measures the load speed of a page for web frameworks.

Runtime Performance The total runtime performance of the application after loading is important. The dynamic page should respond quickly to user interaction. The user interface elements should react without any lag. The animations have to be smooth. These indicators create an impression on the user about the frameworks performance.

14.5 Evaluation

In this section we present the results of our evaluation. We provide the result of each framework in the following subsections, respectively. Table 14.1 gives the summary of the evaluation.

Table 14.1 The evaluation summary of the web frameworks against criterion

Criterion	Ruby on Rails	Django	Spring	CodeIgniter
License and cost	1	1	1	1
Long-term viability	1	1	1	3
Documentation and support	1	1	1	4
Learning success	1	1	3	3
Development effort	2	2	4	3
Modifiability	3	2	2	3
Maintainability	2	2	2	3
User interface	5	5	2	4
Look and feel	4	5	5	5
Load-time performance	3	3	5	1
Run-time performance	2	3	4	1

14.5.1 Evaluation Process

The evaluation process was divided into two phases. In the first phase data and information were collected about the framework to get an initial impression. Online documentations, manuals, and forums were utilized to achieve this. Criteria such as cost and license were assessed in this manner. In the second phase a prototype application, a To-do list, was developed using all the frameworks being tested. Based on this experience, a reviewer rated the frameworks on a scale from 1, excellent, to 6, inferior, for each criterion. In case where the framework selection needs to be done automatically, this information can be automatically acquired and inferred through, for example, crowd-sourcing process, or machine learning (training and prediction or decision-making). The summary of the evaluation can be seen in Table 14.1. Web framework selection is an optimization problem [24] because we need to maximize the goals of the organization. We can measure complex requirement criteria by dividing them into sub-criteria and can use functions such as analytic hierarchy process [31] to evaluate final decision criteria. If the decision process is broken down into smaller manageable components, it results in an improved decision-making. Each of these criteria can be given a weight, according to the goals of an organization. Evaluating frameworks based on a single criterion is difficult and a vague measure. Hence, we need to develop a set of criteria, for the purpose of evaluation of frameworks. The overall goal of the decision criteria is to allow an organization to select an optimal framework based on their requirements. Hence, we divide the criteria into two, first from a developers' viewpoint, in terms of usability and second, from the users' viewpoint. A users' viewpoint and their experience with the application is of paramount importance. From developers' viewpoint, we consider the experience and decisions of the developer, such as licensing and cost. Criteria can be classified in two categories: qualitative and binary. Binary criteria are criteria that can be answered using either using yes or no. It examines whether a feature is available in the framework. Qualitative criteria deals with the quality of the framework. Qualitative criteria are extremely useful for decision-making. Hence this article will deal with qualitative criteria.

14.5.2 Ruby on Rails

Ruby on Rails is released under MIT License, which backs both closed- and open-source projects. It is currently hosted on Github. There is no cost for extra support or other developmental tools. Hence, grade 1 for license and cost. Rails is still in active development and releases new versions and updates frequently. RoR is used by many notable firms such as Twitter, Shopify, and SoundCloud. Rails recently released version 4.2. These positive trends predict a good performance in near future in terms of long-term feasibility. Hence, rank 1.

The documentation provides detailed instructions and tutorial about all available features with detailed examples. Many popular textbooks, articles, and tutorials exist. Forums and stack overflow provides support for Rails. Hence, documentation and support is also evaluated to 1. RoR can easily be learned due to good quality of documentation. It is a highly intuitive language with a highly intuitive syntax, which is very near to the natural language. No extra concepts are required to learn Rails. Hence, we can rank it 1.

Static and dynamic applications can be developed quite easily. Most of the database side code is generated by the scaffold generator. MVC model makes it easier to code. There are many third-party IDEs, even though one is not required. One of the major IDEs is RubyMine from JetBrains. It provides various advanced functionality such as internationalization. It speeds up the development of web application rapidly. Hence, we can provide development effort a grade of very good (2).

Rails is highly modular in nature. It uses third-party plugins called as gems. Any functionality can be added to Rails using third-party gems. As these gems are third party, they are not well documented and hence causes difficulty in extending the software. Hence, extensibility is satisfactory (3). Rails is written using Ruby. An application can be separated to various components in Rails. Since it is based on conventions rather than configuration, Rails is easier to maintain. Therefore, maintainability is very good (2).

Rails by itself does not provide any user interface elements, other than the one supported by HTML, which is not visually appealing. You have to use third-party themes such as Bootstrap or Foundation to provide themes to various UI interfaces. Hence UI elements get a grade of not well fulfilled (5). Rails by default not provide a native look and feel. It will not change from platform-to-platform. Hence it gets a rating of satisfactory (4). The load time of the Rails application depends upon the number of jQuery scripts and CSS classes being loaded. If the pages are highly dynamic, then the performance slightly decreases. This can be alleviated by using minified jQuery. It runs animation fluently once loaded. The performance after loading is excellent. Hence, the load-time performance is satisfactory (3) and the runtime performance is very good (2).

14.5.3 Django

Django uses the 3-clause BSD License also known as modified or revised BSD License which supports both open-source and closed-source development (1). Django is maintained and developed by a non-profit organization called Django Software Foundation. Many major organizations such as Instagram, Mozilla, and Bitbucket use Django for their web application. One of the major goals of the organization is the long-term viability of Django framework. Django is actively developed and frequently released with new updates and bug fixes due to which the Django can be predicted to have a solid future (1).

The Django website provides a good documentation on all of its features along with a separate tutorials for beginners, intermediate, and advanced developers. It provides a good elucidation on all of the APIs provided Django. Thus, Django has an excellent documentation (1). Since Python is based on Python developers will have to learn Python to code, to code in Django. But Python is a simple language to master. But good documentation paves an easy path for learning (1). It has a good code scaffold that does a lot of code scaffolding that reduces the number of lines to be written. Database transactions are automatically handled by an ORM. But the visual elements including CSS and JavaScript has to be coded separately. As a result, the development effort is nearly minimal in Django (2). Django is modular in structure. You can add python packages to Django to extend the functionality. These packages are developed by third-party developers and is hosted at PyPi. Django has pretty straightforward code due to the simplicity of python. Therefore, Django has a very good extensibility (2). Due to modular design and comprehensible code, Django is maintainable (2).

Django doesn't come with any template engine or user interface elements. We need to use third part party templates or custom CSS to provide it exceptional looks. Hence, the look and feel of Django can be rated as poor (5). Moreover, it does not support native look and feel, which would require higher customization using CSS (5). Django is a big framework with a size of 7.5 MB. Applications built with Django tend to be large in size. It provides an extra admin panel which provides a complete control of your application. Django loaded pretty fast (2) and had a very good runtime (2).

14.5.4 CodeIgniter

CodeIgniter is licensed under MIT License which supports both open- source and closed-source software. Earlier it was licensed under Apache/BSD-style open-source license. Due to GPL, incompatibility of the license was shifted to MIT License (1). It is currently maintained and developed by students of the British Columbia University. Hence licensing of CodeIgniter is excellent, but since it is developed in an academic environment without any commercial support its future is not very bright (3).

The documentation of CodeIgniter provides basic coverage of all features and a small tutorial. Initial learning curve is small but later on it becomes quite difficult to master complex development scenarios. The online help is not advanced. There are third-party tutorials and books for mastering CodeIgniter. Hence for long-term feasibility it is satisfactory (4) and documentation and support is average (3). CodeIgniter is based on PHP. It is based on MVC framework. It is easy for the beginners to get started. But mastering requires quite an effort (3). Various integrated development environments are available for the development of PHP such as Zend, NetBeans, and Eclipse. Development effort is low for a simple project, but it becomes harder for complex projects, but it doesn't require any configuration. It is a zero configuration framework. Hence, the net development effort is average (3).

When the project becomes large and complex it becomes difficult to maintain and extend the application mainly due to the fact that PHP consists of spaghetti code. But simple plugins can be added to most of the simple functionalities. Hence, the extensibility is average (3). Maintainability of large and complex applications is going to be a problem. Hence, the maintainability is average (3).

CodeIgniter doesn't provide any user interface elements. It uses the normal elements provided by HTML. Any further modification requires third-party themes that are available as modules. We can use CSS to liven up the application but it requires greater development effort (4). Moreover, it doesn't provide native look and feel. Its look and feel is independent of the platform (5). CodeIgniter is the smallest framework of all the frameworks considered in this test. It just has a size of 2.5 MB. It has a minimal footprint and hence it has fastest loading time (1). Runtime performance is excellent for CodeIgniter (1).

14.5.5 Spring

Spring is licensed under Apache 2.0. Apache 2.0 is a copyright license that is compatible with both open source and closed source, so the license criterion has been fulfilled (1). The Spring software is a huge framework made for enterprises and is usually considered as an alternative for Java Bean. It is currently maintained by Pivotal Software which is a large enterprise that creates software such as VMware. Major release rolls out every year with new features and fixes. Since it is supported by a large corporation, and the code is open source, it has a very strong solid future with high potential (1).

Spring has very detailed and long documentation of all its features. Since it is an enterprise based MVC framework, it has a lot of documentation that needs to be learned. There are online resources and tutorials available. Hence, it provides excellent documentation and support (1). Since it is an enterprise framework, the learning curve for Spring framework is quite high. You will need to be familiar with various APIs and its documentation. It introduces you to some new concepts that are different from normal programming paradigm. We have to learn new concepts such as dependency injection, Java servlets, and JSP, each of which is huge. Spring framework itself consists of a number of modules. Hence, the learning effort is quite high for Spring framework (5). The time taken to develop is also quite high. Spring is not suitable for the development of a small scale application. The development time will span up to more than a year using Spring framework. The number of lines of code required to achieve a certain functionality is much more in Java. As a result, the development time for Spring framework is higher than other frameworks (5).

Java code is divided into classes and packages. Since Spring uses Java it is mostly extensible. You can easily add new classes to Spring framework. But the integration to the framework is not quite easy (3). It is much more difficult to maintain a Spring application due to the complexity involved. As a result, the maintainability of Spring is satisfactory (4). Spring framework provides swing based user interface elements. These elements do not provide native look and feel of the environment. Since Java

is platform independent, the look and feel of the Spring framework is also platform independent. Third-party themes can be used to enhance the visual impact of the applications being developed, CSS and JavaScript. Hence the native look and feel is also satisfactory in Spring framework (5).

Spring MVC has a whopping size of 16 MB, therefore, is a heavy and complex framework with many components. This tends to reflect in its load time. As Java is heavy, clunky, and slow, Spring inherits these qualities from Java and tends to be slower than the other tested frameworks (5). The runtime performance of Spring is also poor. Since Spring uses Java servlets to run and serve web pages, its run-time performance is also slower (4).

14.6 Discussion

In this section, we summarize the weaknesses and strengths of each web framework. We will discuss scenarios and the scenarios in which different frameworks are suitable. Figure 14.3 illustrates the summary of our ranking for the analytical hierarchal process.

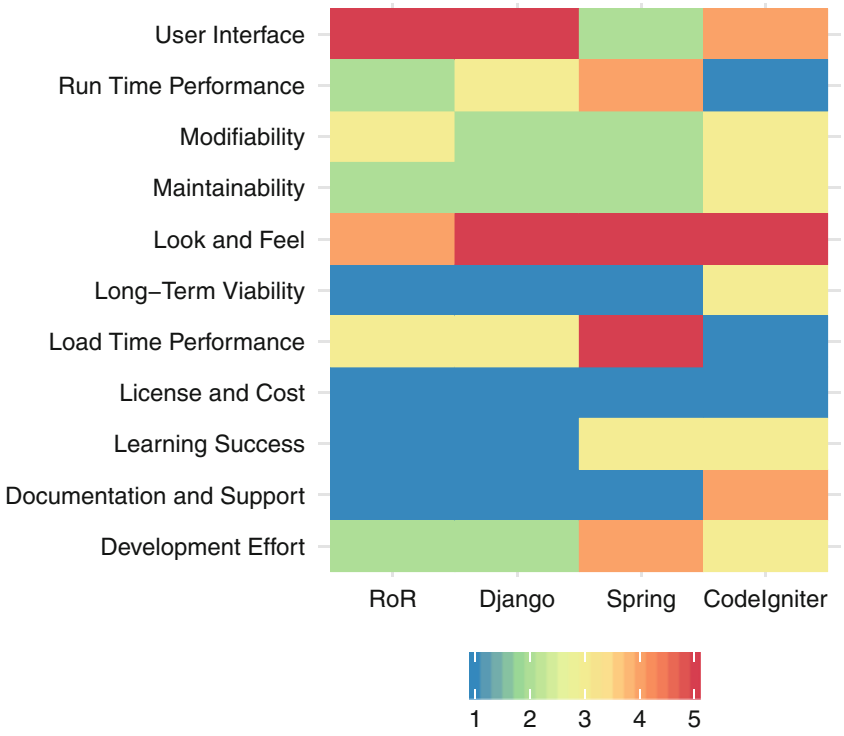


Fig. 14.3 Ranking of web frameworks against criterion on a scale of five

Ruby on Rails is a framework that is suitable for rapid application development. It is a popular software used by startup companies to develop web application due to ease of use and rapid development. Rails concentrates more on the business layer of application development. It provides advanced ORM functionalities. It would have been better if the user interface for Rails was defaulted to bootstrap. A Larger organization with huge applications will have to migrate if the total users and connections increase.

Django could also get some touch up on the UI front. The difference between ROR and Django boils down to the programming languages being used. Other than that, it is almost equivalent. Familiarity with Python or Ruby would be the decider in such scenarios.

CodeIgniter is written in PHP. PHP normally has an unsavory reputation of being spaghetti in style. But they produce ultra-fast, highly minimal applications with a minimal footprint. CodeIgniter can be used to develop applications that require faster response as it has an excellent loading time. It is suitable for smaller application. For larger application, it induces complexity and becomes an overhead.

Spring is an enterprise level framework that can handle data-intensive application. It is not suitable for developing a small application. For smaller applications it increases the development time and the complexity of the code, which degrades the performance of the smaller application, whereas for the larger enterprise solution, it is the best solution as it can handle data-intensive tasks and does handle heavy traffic and sizable connections without crashing the web server.

We have summarized the plus point and negatives of various scenarios, where the frameworks discussed can be used. We should have a weight for different criteria depending upon the requirements of the applications. Table 14.2 is from where you have to begin from where you can use additive principals to select an appropriate framework. In summary, if the web application is small and needs to be

Table 14.2 Results of the WebSelec for the example scenario showing the maximum and minimum for each criterion

	Weight	CodeIgniter	Ruby on Rails	Django	Spring
Select framework	100.00	**38.79**	26.23	20.38	14.60
Load-time performance	26.69	**18.46**	3.58	3.58	1.08
Real-time performance	24.16	**12.95**	6.68	2.99	1.55
Development effort	7.75	**0.77**	3.05	3.05	0.87
Long-term viability	7.35	**0.46**	2.30	2.30	2.30
Look and feel	6.78	1.13	**3.39**	1.13	1.13
Maintainability	6.29	**0.90**	1.80	1.80	1.80
License and cost	5.76	1.44	1.44	1.44	1.44
Documentation and support	5.29	**0.24**	1.68	1.68	1.68
Learning success	4.34	0.54	1.63	1.63	0.54
Modifiability	3.41	**1.15**	**0.54**	0.64	1.08
User interface	2.17	**0.74**	0.15	0.15	1.13

developed faster, then you have the option to select ROR or Django, depending upon the preferred programming languages. If the application is required to be fast, then CodeIgniter is a good option. If the application is being developed for a large organization with large development time frame, then Spring is a suitable framework.

14.7 Example Scenario

In this section, we discuss an example scenario wherein we use our SelecWeb tool for the selection of the best web framework that satisfies the requirements of an example application. The following were the selected requirements for the application in decreasing order of priority:

1. Load-time performance of the web application
2. Real-time performance of the web application
3. Modifiability of the application
4. User interface of the web application

Requirements such as look and feel, development effort, maintenance, and documentation were deemed as negligible or of lower priority. With these requirements, we use the SelecWeb tool to select the best format. The results of the SelecWeb tool are given in Fig. 14.4 and Table 14.2.

	Weight	CodeIgniter	Ruby on Rails	Django	Spring	Inconsistency
Select Framework	100.0%	38.8%	26.2%	20.4%	14.6%	25.6%
Load-Time Performance	26.7%	18.5%	3.6%	3.6%	1.1%	9.0%
Real-Time Performance	24.2%	12.9%	6.7%	3.0%	1.5%	0.8%
Developement Effort	7.7%	0.8%	3.1%	3.1%	0.9%	7.0%
Long-Term Viability	7.3%	0.5%	2.3%	2.3%	2.3%	0.0%
Look and Feel	6.8%	1.1%	3.4%	1.1%	1.1%	0.0%
Maintainability	6.3%	0.9%	1.8%	1.8%	1.8%	0.0%
License and Cost	5.8%	1.4%	1.4%	1.4%	1.4%	0.0%
Documentation and Support	5.3%	0.2%	1.7%	1.7%	1.7%	0.0%
Learning Success	4.3%	0.5%	1.6%	1.6%	0.5%	0.0%
Modifiability	3.4%	1.2%	0.5%	0.6%	1.1%	27.4%
User Interface	2.2%	0.7%	0.1%	0.1%	1.1%	10.1%

Fig. 14.4 The results of the WebSelec for the example scenario

In Fig. 14.4, **Weight** indicates the weight of each requirement that has contributed in the selection of appropriate web framework. We observe that with the given requirements AHP has given a higher score to CodeIgniter (38.8%), followed by Ruby on Rails (38.8%), and Django (38.8%). Higher score indicates a higher achievement of the provided requirements. Hence, with a higher score from our analysis using WebSelec, it is much more feasible to develop the web application using CodeIgniter for the achievement of the requirements. The bold, red values in Table 14.2 indicate the requirements with higher score and bold, blue colored values indicate the lowest score for a given requirement. Requirements such as load-time performance, real-time performance, and modifiability have higher weights for CodeIgniter (18.46% and 12.95%, respectively) as compared to other frameworks, whereas the requirements we assigned least priority such as development effort, long-term viability, maintainability, and documentation and support are lowest in CodeIgniter.

This example clearly illustrates the feasibility and utility of the Analytic Hierarchal Process for selecting a web application framework with multiple requirements or decision makers.

14.8 Conclusion

In this paper, we proposed SelecWeb, an automatic tool for selecting a web framework based on a set of criteria and developer preferences. We presented an analysis of web development framework. A set of criteria was derived, based on application requirements. The set of criteria was used to test and evaluate the web application frameworks. Ruby on Rails, Django, CodeIgniter, and Spring were the web application frameworks that were tested. Each framework was tested by the authors. The assessment and evaluation are valid for the near future but might change as the quality of technology varies. Hence the accuracy of the information is not guaranteed for longer time frame but the methodology and general information provided will remain applicable. Using a case study, we demonstrated the use of the SelecWeb tool to select the best web framework that satisfies the requirements of a given application.

Future work will focus on the security evaluation and detailed performance assessment of the web development frameworks. Moreover, the current development of the tool is based on the AHP method. In the future, we plan to use machine learning techniques to automatically predict the best web development framework for developers and users. The rankings of the web frameworks, based on the discussed criteria (license and cost, long-term viability, etc.), can be used to train a machine learning based model. The trained model will be able to automatically select the best framework based on the given preferences of the users and developers.

Acknowledgements The authors acknowledge with thanks the technical and financial support from the Deanship of Scientific Research (DSR) at the King Abdulaziz University (KAU), Jeddah, Saudi Arabia, under the grant number G-673-793-38. The work carried out in this paper is supported by the High Performance Computing Center at the King Abdulaziz University, Jeddah.

References

1. Muhammed, T., Mehmood, R., Albeshri, A., Katib, I.: UbeHealth: a personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities. IEEE Access **6**, 32258–32285 (2018)
2. Schlingensiepen, J., Nemtanu, F., Mehmood, R., McCluskey, L.: Autonomic Transport Management Systems—Enabler for Smart Cities, Personalized Medicine, Participation and Industry Grid/Industry 4.0., pp. 3–35. Springer, Cham (2016)
3. Mehmood, R., Graham, G.: Big data logistics: a health-care transport capacity sharing model. Procedia Comput. Sci. **64**, 1107–1114 (2015). Conference on ENTERprise Information Systems/International Conference on Project MANagement/Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN / HCist 2015 October 7–9, 2015
4. Mehmood, R., Meriton, R., Graham, G., Hennelly, P., Kumar, M.: Exploring the influence of big data on city transport operations: a Markovian approach. Int. J. Oper. Prod. Manag. **37**(1), 75–104 (2017)
5. Mehmood, R., Lu, J.A.: Computational Markovian analysis of large systems. Int. J. Manuf. Technol. Manag. **22**(6), 804–817 (2011)
6. Mehmood, R., Lu, J.A.: Computational Markovian analysis of large systems. Int. J. Manuf. Technol. Manag. **22**(6), 804–817 (2011)
7. Arfat, Y., Aqib, M., Mehmood, R., Albeshri, A., Katib, I., Albogami, N., Alzahrani, A.: Enabling smarter societies through mobile big data fogs and clouds. Procedia Comput. Sci. **109**, 1128–1133 (2017). 8th International Conference on Ambient Systems, Networks and Technologies, ANT-2017 and the 7th International Conference on Sustainable Energy Information Technology, SEIT 2017, 16–19 May 2017, Madeira, Portugal
8. Suma, S., Mehmood, R., Albugami, N., Katib, I., Albeshri, A.: Enabling next generation logistics and planning for smarter societies. Procedia Comput. Sci. **109**, 1122–1127 (2017). 8th International Conference on Ambient Systems, Networks and Technologies, ANT-2017 and the 7th International Conference on Sustainable Energy Information Technology, SEIT 2017, 16–19 May 2017, Madeira, Portugal
9. Arfat, Y., Mehmood, R., Albeshri, A.: Parallel shortest path graph computations of United States road network data on apache spark. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 323–336. Springer, Cham (2018)
10. Alam, F., Mehmood, R., Katib, I.: D2TFRS: an object recognition method for autonomous vehicles based on RGB and spatial values of pixels. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 155–168. Springer, Cham (2018)
11. Muhammed, T., Mehmood, R., Albeshri, A.: Enabling reliable and resilient IoT based smart city applications. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 169–184. Springer, Cham (2018)
12. Alotaibi, S., Mehmood, R.: Big data enabled healthcare supply chain management: opportunities and challenges. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 207–215. Springer, Cham (2018)
13. Khanum, A., Alvi, A., Mehmood, R.: Towards a semantically enriched computational intelligence (SECI) framework for smart farming. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 247–257. Springer, Cham (2018)

14. Usman, S., Mehmood, R., Katib, I.: Big data and HPC convergence: the cutting edge and outlook. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 11–26. Springer, Cham (2018)
15. Alomari, E., Mehmood, R.: Analysis of tweets in Arabic language for detection of road traffic conditions. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 98–110. Springer, Cham (2018)
16. Suma, S., Mehmood, R., Albeshri, A.: Automatic event detection in smart cities using big data analytics. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 111–122. Springer, Cham (2018)
17. Mehmood, R., Alam, F., Albogami, N.N., Katib, I., Albeshri, A., Altowaijri, S.M.: UTiLearn: a personalised ubiquitous teaching and learning system for smart societies. IEEE Access **5**, 2615–2635 (2017)
18. W3C: HTML5. http://www.w3.org/TR/html5/ (2015). Online; Accessed 20 May 2015
19. Bachle, M., Kirchberg, P.: Ruby on rails. IEEE Softw. **24**(6), 105–108 (Nov 2007)
20. Arthur, J., Azadegan, S.: Spring framework for rapid open source J2EE web application development: a case study. In: Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2005 and First ACIS International Workshop on Self-Assembling Wireless Networks. SNPD/SAWN 2005, pp. 90–95. IEEE, Piscataway (May 2005)
21. Raible, M.: My comparing JVM web frameworks presentation from Devoxx 2010. https://raibledesigns.com/rd/entry/video_of_comparing_jvm_web (2013). Online; Accessed 20 May 2015
22. Rodriguez-Martinez, M., Seguel, J., Greer, M.: Open source cloud computing tools: a case study with a weather application. 2013 IEEE Sixth Int. Conf. Cloud Comput. **0**, 443–449 (2010)
23. Smutny, P.: Mobile development tools and cross-platform solutions. In: 2012 13th International Carpathian Control Conference (ICCC), pp. 653–656. IEEE, Piscataway (May 2012)
24. Heitkötter, H., Majchrzak, T., Ruland, B., Weber, T.: Comparison of mobile web frameworks. In: Krempels, K.H., Stocker, A. (eds.) Web Information Systems and Technologies. Lecture Notes in Business Information Processing, vol. 189, pp. 119–137. Springer, Berlin (2014)
25. Wikipedia: Web application framework — Wikipedia, the free encyclopedia (2015). Online; Accessed 20 May 2015
26. Python Community: WebFrameworks - Python Wiki. https://wiki.python.org/moin/WebFrameworks (2018). Online; Accessed 20 May 2018
27. Ruby, S., Thomas, D., Hansson, D.H.: Agile Web Development with Rails. Pragmatic Bookshelf (2011). ISBN: 1934356549, 9781934356548
28. Wikipedia: CodeIgniter—Wikipedia, the free encyclopedia (2013). Online; Accessed 20 May 2015
29. Stack Overflow: What is dependency injection? http://stackoverflow.com/questions/130794/what-is-dependency-injection (2008). Online; Accessed 20 May 2015
30. Programmers Stack Exchange: Spring introduction and research. http://programmers.stackexchange.com/questions/92393/what-does-the-spring-framework-do-should-i-use-it-why-or-why-not (2011). Online; Accessed 20 May 2015
31. Saaty, T.L.: Axiomatic foundation of the analytic hierarchy process. Manage. Sci. **32**(7), 841–855 (July 1986)
32. Apache: Apache license, version 2.0. http://www.apache.org/licenses/LICENSE-2.0 (2018). Online; Accessed 20-May-2015
33. MIT: The MIT license (MIT)-Open Source Initiative. http://opensource.org/licenses/MIT (2018). Online; Accessed 20 May 2015
34. Sen, R., Subramaniam, C., Nelson, M.L.: Open source software licenses: strong-copyleft, non-copyleft, or somewhere in between? Decis. Support Syst. **52**(1), 199–206 (December 2011)
35. GNU Foundation: Licenses - GNU project - Free Software Foundation. http://www.gnu.org/licenses/ (2018). Online; Accessed 20 May 2015

Part IV
Big Data and High Performance Computing

Chapter 15
On Performance of Commodity Single Board Computer-Based Clusters: A Big Data Perspective

Basit Qureshi and Anis Koubaa

15.1 Introduction

Big data technologies are becoming ever more popular and are currently a focus of both science and industry. The amount of data generated by scientific as well as business applications has increased manifolds in the last few years. A key framework for processing large datasets is the MapReduce framework which allows data to be divided into fixed-size chunks that are processed in parallel on the cloud infrastructure. Several open source MapReduce frameworks have been developed in the last years with the most popular one being Hadoop. Hadoop has been deployed on physical servers across data centers around the globe and continues to provide the realization of on-demand resource availability, scalability with reliability for big data analyses. Figure 15.1 shows the coupling of various technologies for big data analysis in cloud computing infrastructure.

A leading motivation for cloud computing is the reduction of installation and operational cost for small businesses and enterprises. On the other hand, it is immensely important for students in universities to be exposed to real cloud computing infrastructure. Indeed, universities and academic institutions need to provide hands-on experience in this area, which means that universities need

B. Qureshi (✉)
Prince Sultan University, Riyadh, Saudi Arabia
e-mail: qureshi@psu.edu.sa

A. Koubaa
Prince Sultan University, Riyadh, Saudi Arabia

Gaitech Robotics, Shanghai, China

CISTER, INESC-TEC, ISEP, Polytechnic Institute of Porto, Porto, Portugal
e-mail: akoubaa@psu.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_15

Fig. 15.1 Role of cloud infrastructure in big data analysis [2]

to provide access to a suitable cloud computing infrastructure that can be used for experimentation, research, and teaching. Setting up cloud infrastructure in universities could be a very costly endeavor [24]. Although most universities do not reveal the actual costs of setting up and running the infrastructure, the cost of Ukko Cloud Computing Cluster with 240 Dell PowerEdge M610 nodes, each with 32 GB of RAM and 2 Intel Xeon E5540 2.53 GHz quad-core CPUs at University of Helsinki Finland was reported to be over 1 million Euros [13]. Expedient, a private cloud data center construction organization for small businesses, estimates the cost of installation of a tier III data center with ten racks to be upwards of 1 million US Dollars [12].

In order to build a low-cost effective cloud computing cluster with low energy consumption requirements resulting in near-zero carbon footprint, researchers have investigated the use of SBCs. Indeed, an SBC is a complete computer built on a single circuit board that incorporates a microprocessor(s), memory, I/O as well as multitude of other features required by a functional computer [3]. Typically, an SBC is ideally priced at (35–80 US$), with power requirements set to be as low as 2.5 W and designed in small form factors comparable to a credit card or pocket size. These computers are portable and are capable of running a wide range of platforms including Linux distributions, Unix, Microsoft Windows, Android, etc. A cluster of single board computers has very limited resources and cannot compete with the performance of higher value systems. But despite these drawbacks, useful application scenarios exist, where clusters of single board computers are a promising option. This applies in particular to small- and medium-sized enterprises as well as for academic purposes like student projects or research projects with limited financial resources.

The Beowulf cluster created at Boise State University [7] was perhaps the earliest attempt at creating a cluster consisting of multiple nodes of SBCs. This cluster is composed of 32 Raspberry Pi Model B computers and offers an alternative in case if the main cluster is unavailable. The Bolzano Raspberry Pi cloud cluster experiment implemented a 300 node Pi cluster [8]. The main goal of this project was to study the process and challenges of building a Pi cluster on such a large scale.

The Iridis-Pi project implemented a 64 node Raspberry Pi cluster [9]. Tso et al. [10] built a small-scale data center consisting of 56 RPi Model B boards. The Glasgow Raspberry Pi Cloud offers a cloud computing testbed including virtualization management tools. Whitehorn [11] presented the first ever implementation of a Hadoop cluster using five Raspberry Pi Model B nodes. In 2016, C. Baun in [14] presented the design of a cluster geared towards academic research and student scientific projects building an eight-node Raspberry Pi Model 2B cluster. All of these works demonstrate constructing a cluster using SBCs at an affordable cost to researchers and students. However, none of these works provide detailed performance analysis of computing tasks, memory, storage utilization, and network throughput. Indeed effective Hadoop deployment depends on efficient utilization of resources available onboard cluster nodes as well as network traffic management. The lack of performance evaluation of SBC-based cloud computing clusters as well as energy efficiency provides motivation for this work.

In this chapter, we present a detailed study on design and deployment of two SBC-based clusters using Raspberry Pi Model 2 B and HardKernel Odroid Model Xu-4. The objectives of this study are in three folds: (1) To provide a detailed analysis of the performance of Raspberry Pi and Odroid XU-4 SBCs in terms of power consumption, processing/execution time for various tasks, storage read/write as well as network throughput; (2) To study the viability and cost-effectiveness of the deployment of SBC-based Hadoop clusters against virtual machine-based Hadoop clusters deployed on personal computers and (3) To contrast the power consumption and performance aspects of SBC-based Hadoop clusters for Big Data Applications in academic research. To this end, three clusters were constructed and deployed for extensively studying the performance of individual SBCs as well as a cluster deployment to provide a detailed comparison. Furthermore, Hadoop was deployed on these clusters to study the performance aspects of the environment using popular and widely used performance benchmarks. Power consumption, task execution time, I/O read/write latencies as well as network throughput were studied. In addition to the above, we provide analysis of energy consumption in the clusters, the energy efficiency, and cost of operating these clusters. Results from this study show that it is possible to deploy a cost-effective Hadoop cluster with reasonable performance for low yield workloads; however for larger workloads, the operation cost would significantly increase.

The contribution of this chapter is as follows:

- Design and compact layout for two clusters using SBCs are presented in addition to a PC-based cluster running in the virtual environment. Performance evaluation of task execution time, storage utilization, network throughput as well as power consumption are detailed.
- Popular Hadoop benchmark programs such as Pi Computation, Wordcount, TestDFSIO, TeraGen, and TeraSort are executed on these clusters and results are compared against a virtual machine-based cluster using workloads of various sizes.

The remainder of this chapter is organized as follows. Section 15.2 presents related works with details on the ARM-based computing platforms used in this study as well as a review of recent applications of SBCs in high-performance computing and Hadoop-based environments. Section 15.3 presents the design and architecture of the RPi, Xu20, and HDM Clusters used in this study. Section 15.4 deals with a comprehensive performance evaluation study of these clusters based on popular benchmarks. Section 15.5 provides details on the deployment of Hadoop environment on these clusters with a detailed presentation of performance aspects of Hadoop benchmarks for the clusters. Section 15.6 provides summary and discussion followed by conclusions in Sect. 15.7.

15.2 The Single Board Computers

Advanced RISC Machine (ARM) is a family of Reduced Instruction Set Computing (RISC) architectures for computer processors that are commonly used nowadays in tablets, phones, game consoles, etc. [4]. The ARM is the most widely used instruction set architecture in terms of quantity produced [6]. Since October 2011, the ARM has started to support 64-bit address space and instruction set in the ARM v8 architecture. Currently, ARM Cortex cores architecture is popular and widely used in smartphones, single board computers, etc. An SBC is a complete computer built on a single circuit board. An SBC incorporates a microprocessor(s), memory, I/O as well as host of other features required by a functional computer. While keeping the manufacturing costs to the lowest (25–80 US$), various companies have developed SBCs in small form factors comparable to a credit card or pocket size. These computers are capable of running a wide range of platforms including Linux distributions, Unix, Microsoft Windows, Android, etc. In what follows, we briefly describe the two popular SBCs using ARM-based CPUs and their features.

The Raspberry Pi Model 2B The Raspberry Pi Foundation [1] developed a credit card-sized SBC called Raspberry Pi (RPi). This development was aimed at creating a platform for teaching computer science and relevant technologies at the school level. Raspberry Pi 2B version was released in February 2015 improving the previous development platform by increased processor speed, larger onboard memory size as well as newly added features. Figure 15.2 shows RPi Model 2B. Table 15.1 summarizes the hardware specifications of RPi Model 2B. Although the market price, as well as the cost of energy consumption of an RPi, is low, the computer itself has many limitations in terms of shared compute and memory resources. Raspberry Pi uses a 32-bit quad-core ARM Cortex A7 processor clocked at 0.7 GHz with 256 KB L2 cache memory, which is shared with the GPU. While it is possible to overclock the processor and tune the performance, the results may reduce the overall lifespan of the computer. For data storage, RPi relies on solid state flash memory. The SD memory reads and writes in 128 KB blocks of data, i.e., even for reading/writing one byte, the entire block of memory needs to be read from

Fig. 15.2 Raspberry Pi 2 B

Table 15.1 Features of Raspberry Pi Model 2B and HardKernel Odroid Xu-4

	RPi Model 2B	Odroid XU-4
Processor (CPU)	0.9 GHz quad core ARM Cortex-A7	Samsung Exynos5 Octa ARM Cortex-A15 (@ 2.0 GHz) and Cortex-A7 (@1.3 GHz) CPUs
GPU	Broadcom Video Core IV Multimedia Graphics co-processor	Mali T628 Open GL 3.0
Onboard RAM	256 KB L2 cache 1 GB SDRAM at 400 MHz	2 GB LPDDR3 at 933 MHz
Ethernet/Network	10/100 MB Ethernet RJ45 Jack	10/100/1000 MB Ethernet RJ45 Jack
Storage	Micro SD Card	Micro SD Card and eMMC 5.0 flash storage
Audio/Video	3.5 mm jack and HDMI	HDMI (standard) supports 1080p video
Power Consumption	3.2 W (idle) 3.8 W (under load)	2.5 W (idle) 4.5 W (under load)
USB Ports	4 USB 2.0	1x USB 2.0, 2× USB 3.0
Released	February 2015	2015
Price (US$)	35$	79 $

or written to. Furthermore, the lifespan of the SD card is reduced significantly with very frequent write operations. In summary, the RPi is a very affordable platform with low cost and low energy consumption [3, 4]. The major drawback is the compute performance. Recent experiments in distributed computing have shown that this can be rectified by building a cluster of many RPi computers. Further details about configuration in the cluster would be provided in the next section.

The Hardkernel Odroid platform ODROID-XU-4 [5] is a newer generation of single board computers offered by HardKernel. Offering open source support, the

Fig. 15.3 Hardkernel Odroid XU-4

board can run various flavors of Linux, including Ubuntu 15.04, Ubuntu MATE, Android 4.4 Kit Kat, and 5.0 Lollipop. XU-4 uses Samsung Exynos5 Quad-core ARM Cortex™-A15 Quad 2 GHz and Cortex™-A7 Quad 1.3 GHz CPUs with 2 Gbyte LPDDR3 RAM at 933 MHz. The Mali-T628 MP6 GPU supports OpenGL 3.0 with 1080p resolution via standard HDMI connector. Two USB 3.0 ports, as well as a USB 2.0 port, allows faster communication with attached devices. The power-hungry processor demands 4.0 A power supply with power consumption of 2.5 W (idle) and 4.5 W (under load). By implementing the eMMC 5.0, the ODROID C1 and XU-4 boast improved I/O transfer speeds over Class 10 SD card flash memories. XU-4 comes with an onboard heat sink as well as a fan. With heavy computation loads, the temperature can increase resulting in increased power consumption due to cooling. We noticed that the temperature doubled under increased computation stress resulting in the constant running of the fan creating excessive noise. Odroid XU-4 priced at $79 is slightly expensive compared to Raspberry Pi 3B; nevertheless, the improved processing power although demanding more power provides tradeoff with improved performance, task execution time as well as better I/O read and write operations. Table 15.1 shows a summary of Odroid XU-4 SBC (Fig. 15.3).

The low-cost aspect of an SBC makes it attractive for students as well as researchers in academic environments. As pointed out in the literature, it is possible to deploy a Hadoop cluster using SBCs such as Raspberry Pi computers. Although the Raspberry Pi computers are cheap and widely available, the limitations in terms of processing power, available onboard memory and reliance on SD cards for external storage with slow I/O operations, yield performance with much to be desired. Thanks to increased interest in SBCs, newer single board computers with better design and faster operations speeds are becoming available. It remains to be seen how the improved SBCs perform when deployed in Hadoop clusters. In this chapter, we present a detailed study on design and deployment of Hadoop on two SBC-based clusters using Raspberry Pi Model 2 B as well as HardKernel Odroid

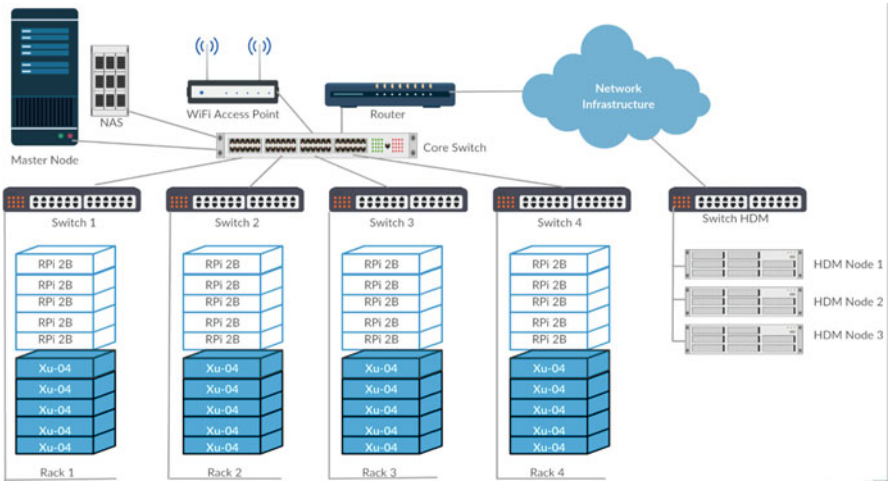


Fig. 15.4 Network topology diagram for RPi, Xu20, and HDM clusters

Model Xu-4. The Odroid XU-4 is an SBC with the faster processor, larger onboard memory, and faster I/O storage.

15.3 Design and Architecture of the DM-Clusters

This section presents the architecture and configuration of the clusters deployed in this experimental study. For the purpose of benchmarking cluster performance as well as comparatively analyzing their performance, we built three clusters.

The first cluster, called RPi Cluster, is composed of 20 Raspberry Pi Model 2B Computers connected to a network. The second cluster, called Xu-20, is composed of 20 Odroid XU-4 devices in the same network topology. The third cluster HDM is composed of four regular PCs running Ubuntu in the virtual environment using VMware Workstation [28]. To maintain similarity in network configuration, all the clusters follow the same star topology with a 24-port Giga-bits-per-second smart managed switch acting as the core of the network as can be seen in Fig. 15.4. Each node (RPi, XU-4, or PC) connects a 16-port Ethernet switch that connects to the core switch. Currently, five nodes connect to each switch allowing further scalability of the cluster. The master node, as well as the uplink connection to the Internet through a router, is connected to the core switch. The current design allows easy scalability with up to 60 nodes connected in the cluster that can be extended up to 300 nodes. Table 15.2 presents a summary of the cluster characteristics.

Table 15.2 Configuration of the DM-Clusters

	RPi Cluster	Xu20 Cluster	HDM Cluster
Master Node	Intel i7 at 3.00 GHZ 64Bit Win 10	Intel i7 at 3.00 GHZ 64Bit Win 10	Intel i7 at 3.00 GHZ 64Bit Win 10
Number of Data Nodes	20	20	4
Slave Node Device	Raspberry Pi Model 2 B	HardKernel Odroid Xu-4	Intel i7 at 3.00 GHZ 64Bit Win 10
Data Node Clock Speed	1000 MHz	2000 MHz	3000 MHz
OS	Raspbian OS	Ubuntu MATE 15 OS	Ubuntu 14.4 LTE
Storage (GB)	16 GB	32 GB	40 GB
Storage Medium	Class 10 SD Card	eMMC 5.0 module	Kingston Solid State Disk (SSD)
RAM	856 MB (available)	1024 MB (available)	3 GB (available)
Virtual Machine	Only Master Node runs OS in VM		All nodes on VM

15.3.1 Components and the Design of the DM-Clusters

Each cluster is composed of a set of components including SBCs, power supplies, network cables, storage modules, connectors, and cases. Each SBC is carefully mounted with storage components. All the Raspberry Pi computers are equipped with 16 GB Class-10 SD cards for primary bootable storage. The Odroid XU-4 devices are equipped with 32 GB eMMCv5.0 modules and can be seen in Fig. 15.3. All the SBCs are housed in a compact layout racks using M2/M3 spacers, nuts, and screws. The racks are designed to house 5 SBCs per rack for easy access and management. Figure 15.5a shows the Raspberry Pi computers organized in racks with 5 computers per rack, Fig. 15.5b shows the Odroid XU-4 computers organized in racks with 5 computers per rack.

Currently, each Raspberry Pi computer is individually supplied by the 2.5 A power supply; each Odroid XU-4 computer is supplied by a 4.0 A power supply that provides ample power for running each node. All the power supplies are connected to the Wattsup Pro .net power supply meter for measuring power consumption. These power meters are then connected to a voltage regulator connected to the main supply. The Wattsup Pro .net power meter can be seen in Fig. 15.6a.

Each SBC's network interface is connected to a Cat6e Ethernet cable through the RJ-45 Ethernet connector. All Ethernet cables connect to the 16-port Cisco switches which connect to a Gigabit Core switch. An Internet router, as well as the Master PC running Hadoop namenode, is connected to the network. Figure 15.6b shows the network connectivity. The HDM Cluster is composed of four PCs all connected in the same network topology as of the other clusters. Each PC is equipped with an Intel i7 4th Gen Processor with 3.0 GHz Clock speed, 8 GB RAM, and 120 GB Solid State Disk Drive for storage. Each PC is equipped with a 400 W power supply

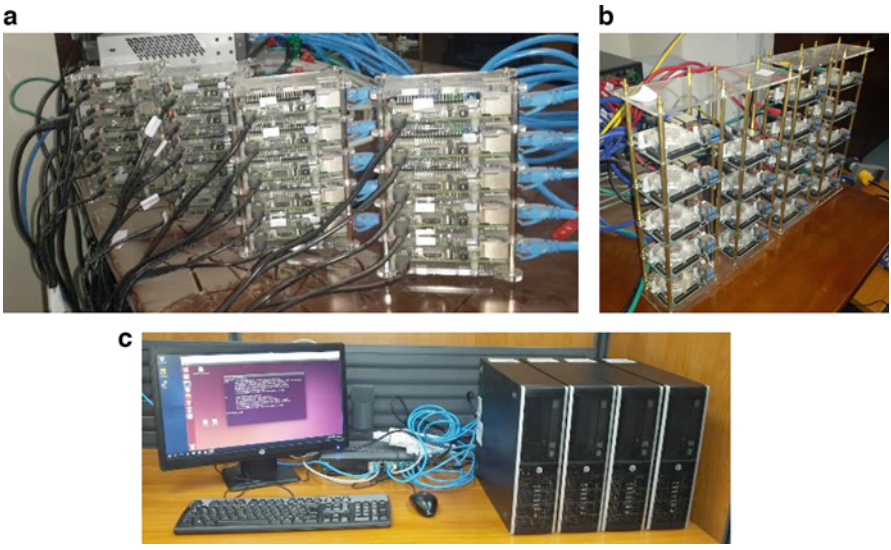


Fig. 15.5 Hardware installation; (**a**) The RPi Cluster composed of 20 RPi Model 2B computers; (**b**) The Xu20 Cluster composed of 20 Odroid XU-4 computers; (**c**) The HDM Cluster composed of 4 Intel 7, 3.0 GHz PCs

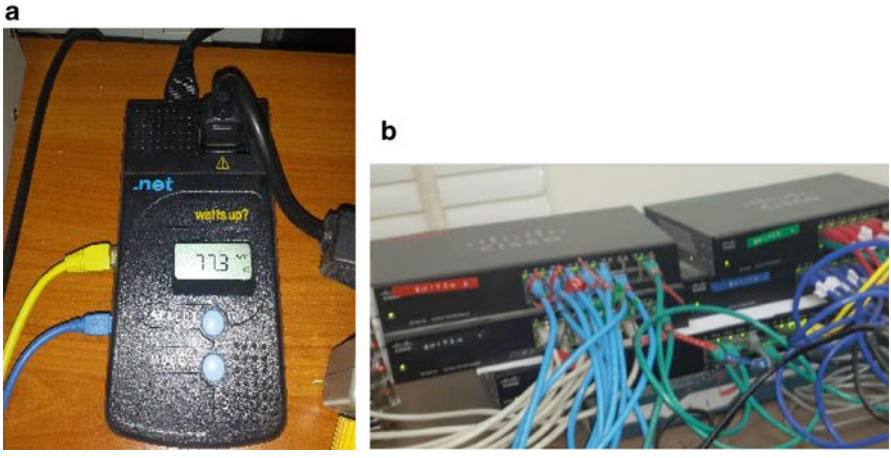


Fig. 15.6 (**a**) Wattsup Pro .net power meter (**b**) Cisco Core switch, Cisco Internet Router, and 4 × 16 port switches

and connects to the Ethernet Switch. Figure 15.5c shows the HDM Cluster. The purchase cost of all components of the RPi, Xu20, and HDM Clusters was $1300, $2700, and $4200, respectively. The Network and Power reading equipment cost is approximately $450.

15.3.2 Raspbian and Ubuntu MATE Image Installation

For the RPi Cluster, we built the RPi Image. The Raspbian OS image is based on Debian that is specifically designed for ARM processors [29]. Using Raspbian OS for RPi is easy with minimal configuration settings requirements. Each individual RPi is equipped with a SanDisk Class 10, 16 GB SD card capable of up to 45 MB/s read as well as up to 10 MB/s write speeds available at a cost of US$15. We created our own image of the OS which was copied on the SD cards. Additionally, Hadoop 2.6.2 is installed on the Image with Java JDK 7 for ARM platform. When ready, these SD cards are plugged into the RPi systems and mounted. The Master node is installed on a regular PC running an Ubuntu 14.4 virtual machine on Windows 10 as the host operating system.

For the Xu20 Cluster, we built another image based on Ubuntu MATE 15.10. Ubuntu MATE is an open source derivate of the Ubuntu Linux distribution with MATE desktop. HardKernel provides Ubuntu MATE 15.10 pre-installed on the Toshiba eMMCv5.0 memory module which is preconfigured for Odroid XU-4 single board computers at a price of US$43. The eMMCv5.0 is capable of reading and write speeds of 140 MB/s and 40 MB/s, respectively. Apache Hadoop 2.6.2 along with Java JDK 7 for ARM platform was installed on the image. These modules were inserted into eMMC socket on the Odroid XU-4 boards and connected to the network. Similar to the RPi Cluster, the Hadoop master node was installed on a regular PC running Ubuntu 14.4 VM.

The final cluster HDM is composed of four PCs all connected in the same network topology as of the other clusters. A virtual machine in the VMware workstation was built to run Hadoop 2.6.2 with Java JDK 7 for 64-bit architecture. One of the VMs serves as the master node and runs Hadoop namenode only. The rest of the VM run the data nodes of the cluster.

15.4 Performance Evaluation of DM-Clusters

In this section, we present a performance evaluation study of DM-Clusters in terms of energy consumption, processing speed, storage read/write, and networking.

15.4.1 Energy Consumption Approximation

Energy consumption in data centers is a major concern for green cloud computing research. The Greenpeace [26] in 2012 estimated the global energy consumption for data centers to be over 31 GW. Recently, the NRDA [27] estimated in 2013, in the USA alone, the data centers consumed 91 billion kiloWatts hours (kWh) of energy, which is estimated to increase by 141 billion kWh every year until 2020, costing

Table 15.3 Power consumption of clusters in idle and stress modes with power cost per year

	Idle mode		Stress mode	
	Power consumption (E) (W)	Power cost in USD	Power consumption (E) (W)	Power cost in USD
RPi CLUSTER (20 NODES)	34.1	$ 14.94	46.4	$ 20.33
XU20 CLUSTER (20 NODES)	56.2	$ 24.63	78.7	$ 34.49
HDM CLUSTER (4 NODES)	108.4	$ 47.51	197.7	$ 86.66

Table 15.4 CPU execution time (s) for individual nodes with n threads

	CPU cores	Clock rate GHz	CPU execution time with n threads				
			1	2	4	8	16
Raspberry Pi 2B	4	1.0	448.2	225.1	113.8	113.7	113.7
Odroid Xu-4	8	2.0	83.3	41.68	25.33	17.66	18.02
Intel i7 4th Gen	4	3.0	8.51	4.272	2.22	2.27	2.23

businesses $13 billion annually in electricity bills and emitting nearly 100 million metric tons of carbon pollution per year. Resource over-provisioning and energy non-proportional behavior of today's servers [25] are two of the most important reasons for high energy consumption of data centers. On the other hand, use of low-end computers is increasingly becoming popular due to low cost and low energy consumption. In this section, we analyze the power consumption of SBCs used in this study.

The energy consumption for the DM-Clusters was measured using the Wattsup Pro .net power meters. These meters provide consumption in terms of Watts for 24 h a day and log these values in local memory for accessibility. To estimate the approximate power consumption over a year, we measured the power consumption in two modes, Idle mode and stress mode for each DM-Cluster. In idle mode, the clusters were deployed without any application/task running for a period of 24 h. In stress mode, the clusters ran a host of computation intensive applications for a period of 24 h. Observing the logs, the upper-bound wattage usage within a period of 23 h was taken as power consumption in the idle mode as well as the stress mode. Table 15.3 shows the power consumption for DM-Clusters in idle and stress modes.

The cost of energy for the cluster is a function of power consumption per year and the cost of energy per kiloWatts hour [23]. An approximation of energy consumption cost per year (C_y) can be given by Eq. (15.1) where E is the specific power consumption for an event for 24 h a day and 365.25 days per year. The approximate cost for all the clusters computed based on values given in Table 15.4, whereas the cost per kilowatt-hour (P) is assumed to be 0.05 US$.

$$C_y = E \times 24\frac{\text{hour}}{\text{day}} \times 365.25\frac{\text{days}}{\text{year}} \times \frac{P}{\text{kWh}} \tag{15.1}$$

The Bolzano Experiment [8] reports Raspberry Pi cluster built using Raspberry Pi Model B (first generation) where each node is consuming 3 W in stress mode. In RPi Cluster, the Raspberry Pi Model 2B consumes slightly less power with 2.4 W in stress mode. We observe that this slight difference in power consumption is due to the improved design of the second-generation Raspberry Pi. The Cardiff Cloud testbed reported in [30] compared two Intel Xeon-based servers deployed in the data center with each server consisting of 2 Xeon e5462 CPU (4 cores per processor), 32 GB of main memory, and 1 SATA disk of 2 TB of storage each. The researchers in this study used similar equipment to measure power consumption as presented in this study. Their work reports that each server on average consumes 115 W and 268 W power in idle and stress modes, respectively. The power consumption for the RPi Cluster with 20 nodes is 5 times better compared to a typical server in a cluster.

In a scenario where the RPi Cluster runs an application in stress mode (i.e., 46.4 W) for the whole year, the cost for power usage is approximately $20.33. For Xu20 and HDM Clusters, the yearly cost would be $34.49 and $86.66, respectively. It is clear that using low-cost low-power devices enable a greener computing environment in terms of energy consumption.

15.4.2 CPU Performance

In this section, we analyze the performance of the DM-Clusters using various benchmark. The objective of this study is to investigate and compare the processing speed of the three platforms under consideration to understand their intrinsic performance.

The benchmark suite Sysbench[1] was used to measure the CPU performance. Sysbench provides benchmarking capabilities for Linux and supports testing CPU, memory, File I/O, mutex performance in clusters. We execute the Sysbench benchmark[2] testing each number up to value 10,000 if it is a prime number for n number of threads [22]. Since each computer has a quad-core processor, we run the sysbench CPU test for 1, 2, 4, 8, and 16 threads. We measure the performance of this benchmark test for Raspberry Pi Model 2B, Odroid XU-4 as well as Intel i7 fourth-generation computers used in the three DM-Clusters. Table 15.4 shows the average CPU execution time for nodes with n threads.

As can be seen from Fig. 15.7, all the tested devices had four cores, the CPU execution times scale well with the increased number of threads. Sysbench test runs

[1] https://wiki.gentoo.org/wiki/Sysbench

[2] Using `sysbench --test=cpu --cpu-max-prime=10000 --num-threads=`*n* `run`

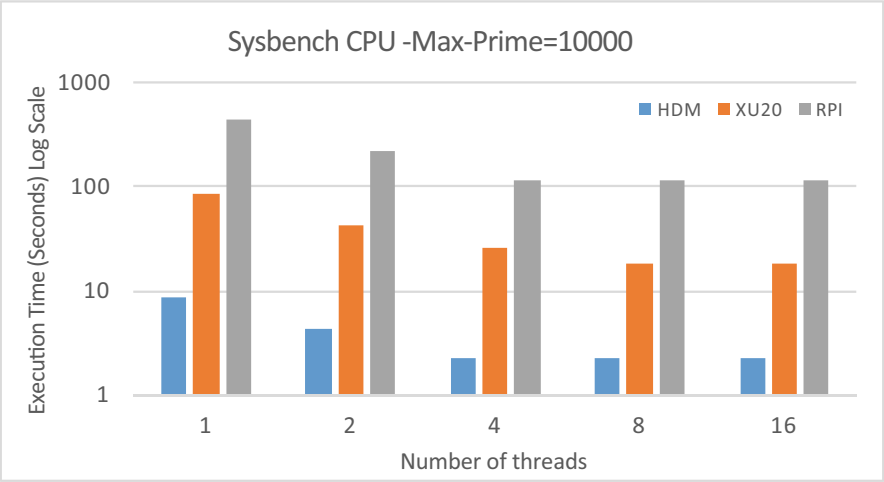


Fig. 15.7 Sysbench CPU execution times for SBCs (logarithmic scale)

with $n = 2$ and $n = 4$ threads significantly improve the execution times performance for all processors by 50%. With $n = 8$ and $n = 16$ threads, the test results yield almost similar execution times with little improvement in performance. It can also be noted from Fig. 15.7 that the execution times for Odroid XU-4 are 10 times better as compared to Raspberry Pi Model 2B. The increased number of threads does not provide gain in performance of Odroid XU-4 over Raspberry Pi; furthermore, the execution time for Raspberry Pi is further extended with larger n. The HDM Cluster nodes run 4.42 times faster compared to Odroid Xu-4. These results clearly illustrate the handicap of SBC onboard processors when compared to a typical PC.

The Raspberry Pi Model 2B allows the user to overclock the CPU rate to 1200 MHz, in our experiments with the over-clocked CPU we did not observe significant improvement using the sysbench benchmark.

15.4.3 Storage Performance

Poor storage read/write performance can be a bottleneck in clusters. Compared to server machines, an SBC is handicapped in terms of availability of limited storage options. SBCs are typically restricted to external storage connected through the USB interface with bootable flash disks or SD cards are primary storage devices. In this section, we compare the storage performance of the DM-Clusters nodes and analyze the performance of three different mediums for storage.

The small scale of the SBCs of Odroid Xu-4, as well as Raspberry Pi Model 2B, provides few options for external storage. Both SBC is equipped with SD Card Memory slots that come with bootable versions of Linux distributions. In addition

Table 15.5 Read and write throughput (KB/s) for individual devices in the clusters using FIO[a]

	Read throughput (KB/s)		Write throughput (KB/s)	
	Buffered	Non-buffered	Buffered	Non-buffered
Raspberry Pi 2B with 16 GB Class 10 SanDisk SDCard	7135	4518	2701	2537
Odroid Xu-4 with 32 GB eMMCv5.0 Module	14,318	13,577	6421	5118
Intel i7 4th Gen with 120 GB SanDisk Solid State Disk	164,521	93,608	96,987	62,039

[a]Measured using fio –name = randread –ioengine = libaio –iodepth = 1 –bs = 4 k –size = 512 M –runtime = 240

to the SD Card Memory slot, the Odroid XU-4 is also equipped with eMMCv5.0 connector. Apart from these, both devices are equipped with USB 2.0 interfaces with Raspberry Pi having 4, XU-4 having only one. The XU-4 is also equipped with two USB 3.0 ports for faster data transfer. Additional storage devices can be mounted using these USB ports. The Raspberry Pi's were equipped with 16 GB SanDisk Class 10 SD cards, whereas the XU-4 devices were equipped with 32 GB eMMC memory cards. Both of these memory cards were loaded with bootable Linux distributions. For comparison purposes, we used 128 GB SanDisk Solid State Disks on the HDM Cluster machines and used flexible IO (FIO) which is commonly used to benchmark IO performance of storage in various Linux distributions.

FIO[3] allows benchmarking of sequential read and write as well as random read and write with various block sizes. NAND memory is typically organized in pages and groups with sizes 4, 8, or 16 Kilobytes. Although it is possible for a controller to overwrite pages, the data cannot be overwritten without having to erase it first. The typical erase block on SD cards is typically 64 or 128 KB. In newer SD cards, the small number of erase blocks are combined into larger allocation units or segments with a size 4 MB. The controllers of the SD cards implement a translation layer maintaining the mapping and translation of virtual and physical memory addresses. As a result of these design features, the random read and write performance of SD cards depends on the erase block, segment size, the number of segments, and controller cache for address translations.

Table 15.5 shows the comparison of buffered and non-buffered random read and write from all the three devices with block size 4 KB. FIO was used to measure the random read and write throughput with eight threads each working with a file of size 512 MB with a total 4 GB of data. These parameters were set specifically to avoid buffering and caching in RAM issues which are managed by the underlying operating systems that can distort the results, i.e., the data size (4 GB) selected is larger than the onboard RAM available on these devices. As can be seen from Table 15.5, the read throughput (buffered) of Odroid with eMMC memory is at least twice as fast as the Class 10 SD card on the Raspberry Pi whereas the non-buffered

[3]https://www.openhub.net/p/fio

read is more than three times better. Similarly, for buffered write operations, Odroid XU-4 with eMMC module throughput is more than twice better when compared to the Class 10 SD card in Raspberry Pi. Table 15.5 also shows the comparison of the throughput of the SSD Storage on the PC in the HDM Cluster against the throughput of these devices. The buffered read throughput for SSD storage is at least 10 times better compared to eMMC module in Odroid XU-4 computers whereas the buffered write throughput of SSD storage is 15 times better. These experimental observations clearly imply the benefit of using SSDs with higher throughput when compared to Class 10 SD cards as well as eMMC v5.0 memory modules. When deployed in a distributed environment such as Hadoop that requires frequent read and write operations, the SD cards with slower read/write throughput can increase the task completion rate. On the other hand, faster memories such as eMMC or SSD Drives can have a pivotal role in improving performance for the applications.

15.4.4 Network Performance

When data are being processed in a cluster, servers need to transfer data with a certain amount of network bandwidth for the data to be delivered quickly and processed efficiently. If the network cannot allocate bandwidth properly, the speed of delivering and processing data will suffer because of unnecessary network congestion among many other reasons. Major factors that can have an impact on data processing and task execution time includes not only the speed of CPU, size of main memory, the speed of storage I/O, but also the allocation of network resources. Figure 15.4 shows the network topology for various networking components in the three clusters. In this section, we provide the comparative analysis of network performance using network throughput and latency using various payload sizes of data over the TCP protocol using Linux-based benchmark tools.

The network performance was measured using the popular Linux-based command line tool iperf v3.13 with the NetPIPE benchmark version 3.7.2. Through various sets of runs, iperf states the network throughput to be 82–88 Mbits per second for the RPi and XU20 Clusters. NetPIPE [15, 16], on the other hand, provides more details considering performance aspects for network latency, throughput, etc. over a range of messages with various payload size in bytes. For this study, we executed the benchmark within the clusters for various payload sizes over the TCP end-to-end protocol. The NPtcp, NetPIPE benchmark using TCP protocol, involves running transmitter and receiver on two nodes in the cluster. In our experimentation, we executed the receiver on the cluster namenode with 1000 KB as maximum transmission buffer size for a period of 240 ms. The transmitter was executed on the individual SBCs one by one.

As can be seen from Fig. 15.8, the network latency for all clusters with small payload is almost similar. As the payload increases, we observe a slight increase in network latency between the three clusters. On the other hand, we observe a spike

in throughput at message size 1000 bytes; this indicates that the smaller a message is, the more is the transfer time dominated by the communication layer overhead.

For larger messages, the communication rate becomes bandwidth limited by a component in the communication subsystem that may include the data rate at the network link, utilization of the communication medium at the time, or the traffic on the network switch. In the context of Hadoop installation in the cluster, the namenode frequently communicates with data nodes using heartbeat messages with smaller payloads, whereas the data blocks typically larger than the 128 MB need to be copied from one data node to another. We present detailed network performance using Hadoop benchmarks in the next section.

We also note that the throughput at the HDM Cluster is lowest compared to the other clusters, this is mainly due to the proximity of the HDM Cluster. This cluster is physically located in a farther area and requires an extra switch to connect to the namenode of the clusters. The physical proximity and the longer distance yields degradation in throughput performance for the HDM Cluster. Contrasting the performance of XU-4 and RPi SBCs, we note the visible difference in throughput between the two, this is due to the poor overall Ethernet performance of the Raspberry Pi probably caused by design. On the Raspberry Pi, 10/100 Mbps Ethernet controller is a component of the LAN9512 controller which contains the USB 2.0 hub as well as the 10/100 Mbit Ethernet controller. On the other hand, the Odroid XU-4 is equipped with an onboard Gigabit Ethernet controller which is part of the RTL8153 controller. The coupling of faster Ethernet port with high-speed USB 3.0 provides better network performance. Figure 15.8 shows comparatively the throughput on the Xu20 Cluster is 1.52 times better when compared to the RPi Cluster.

15.5 Performance of Hadoop Benchmark Tests on Clusters

Apache Hadoop is an open source framework that provides distributed processing of large amounts of data in a data center. The Hadoop framework scales well for thousands of machines allowing processing of petabytes of data. It offers high availability options for detection and recovery from failures in software as well as hardware thus making it a very reliable distributed ecosystem. Hadoop uses the map/reduce programming model for big data processing over multiple nodes. The map/reduce model is composed of two steps, the map step performs filtering and sorting of data, the reduce step provides further processing of data from map step usually summarizing the outcomes. Depending on the application, the map/reduce tasks can be parallelized. Hadoop 2 introduced Yet Another Resource Negotiator (YARN) as a new resource management layer allowing for better resource management and monitoring.

On all three clusters, Hadoop version 2.6.2 was installed due to the availability of YARN daemon which improves the performance of the map/reduce jobs in the cluster. To optimize the performance of these clusters, yarn-site.xml and Mapred-

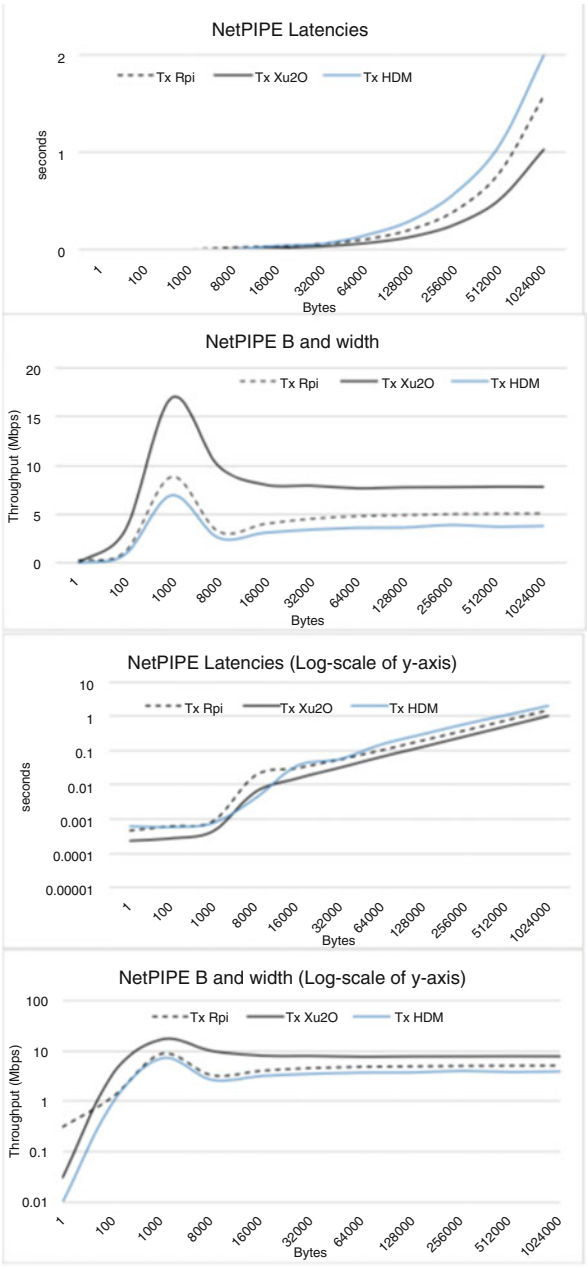


Fig. 15.8 NetPIPE benchmark results for all clusters considering latencies and bandwidth with data size in terms of bytes on the *x*-axis

Table 15.6 Properties in mapred-site.xml

Property	Value
yarn.app.mapreduce.am.resource.mb	852
mapreduce.map.cpu.vcores	1
mapreduce.reduce.cpu.vcores	1
mapreduce.map.memory.mb	852
mapreduce.reduce.memory.mb	852
mapreduce.input.fileinputformat.split.minsize	8 MB

Table 15.7 Properties in YARN-site.xml

Property	Value
yarn.nodemanager.resource.memory-mb	1024
yarn.nodemanager.resource.cpu-vcores	1
yarn.scheduler.minimum-allocation-mb	256
yarn.scheduler.maximum-allocation-mb	852
yarn.scheduler.minimum-allocation-vcores	1
yarn.scheduler.maximum-allocation-vcores	1
yarn.nodemanager.vmem-pmem-ratio	2

Table 15.8 Properties in hdfs-site.xml

Property	Value
dfs.replication	2

site.xml were configured with 852 MB of resource size allocation. The primary reason for this is the limitation in the RPi Model 2B which has 1 GB of onboard RAM out of which 852 MB is available; the rest is used by the Operating System as well as the CPU Memory Bus. The default container size on the Hadoop Distributed File System (HDFS) is 128 MB. Each SBC node was assigned a static IPv4 address based on the configuration and all slave nodes were registered in the Master node. YARN and HDFS containers and interfaces could be monitored using the web interface provided by Hadoop. Tables 15.6, 15.7, and 15.8 provide details of important configuration properties for the Hadoop environment. It must be noted that maximum memory allocation per container is 852 MB; this is set on purpose so that the performance of all clusters could be measured and contrasted. Additionally, the replication factor for HDFS is 2 which means only two copies of each block would be kept on the file system.

These clusters were tested extensively for performance using Hadoop benchmarks for Quasi-Random Pi generation and word count applications.

15.5.1 The Pi Computation Benchmark

Hadoop provides its own benchmarks for performance evaluation over multiple nodes. One of the simplest benchmarks is the computation of the value of π using Quasi-Monte Carlo Method and map/reduce. We execute the compute Pi program

Table 15.9 CPU execution times for Pi computation benchmark on clusters

Map tasks	Samples	Average CPU execution times (s)		
		RPi Cluster	Xu20 Cluster	HDM Cluster
10	10^3	98.469	37.37	22.86
10	10^4	99.13	37.69	20.5
10	10^5	97.902	36.97	18.92
10	10^6	100.629	37.87	25.35
100	10^3	465.675	49.62	17.84
100	10^4	461.4	49.7	19.35
100	10^5	470.264	49.43	20.12
100	10^6	486.48	49.89	21.24

that computes exact m binary digits of the mathematical constant π using a quasi-Monte Carlo method and MapReduce. The precision value m is provided at the command prompt with values ranging from 1×10^3 to 1×10^6 increased at an interval of 1×10^1. Each of these is run against a number of map tasks set at 10 and 100. We study the impact of the value of m versus the number of map tasks assigned and compute the difference in time consumption (execution time) for completion of these tasks. Each experiment is repeated at least 10 times for significance of statistical analysis. In this experimentation, the Pi computation benchmark's goal is to observe the CPU bound workload of all the three clusters. Table 15.9 shows average CPU execution times for various runs of the Pi computation program with 10 and 100 map tasks. Figure 15.9a, b show the box-whisker plot with upper and lower quartiles for each sample set with 10 and 100 map tasks. With 10 maps, the average execution time for RPi Cluster with 10 + E06 number of samples is 100.8 s, whereas for XU20 and HDM Cluster the average execution time is 38.2 and 25.1 s, respectively. As the number of maps increases to 100, we observe significant degradation in performance of RPi Cluster with average execution time at 483.7 s for 10 + E06 number of samples. Comparatively, the execution times for Xu20 and HDM Clusters are 50.1 and 21.8 s, respectively. This clearly shows the significant difference in the computation performance between the RPi Cluster and the Xu20 Cluster. Figure 15.9c shows the ratio of performance degradation of RPi and XU20 Clusters compared to HDM Cluster for Pi program CPU execution times with 10 and 100 maps.

15.5.2 The Wordcount Benchmark

The Wordcount program contained in the Hadoop distribution is a popular micro-benchmark widely used in the community [15]. The Wordcount program is representative of a large subset of real-world MapReduce jobs extracting a small amount of interesting data from a large dataset. The **Wordcount** program reads text files and counts how often words occur within the selected text files. Each mapper takes a line

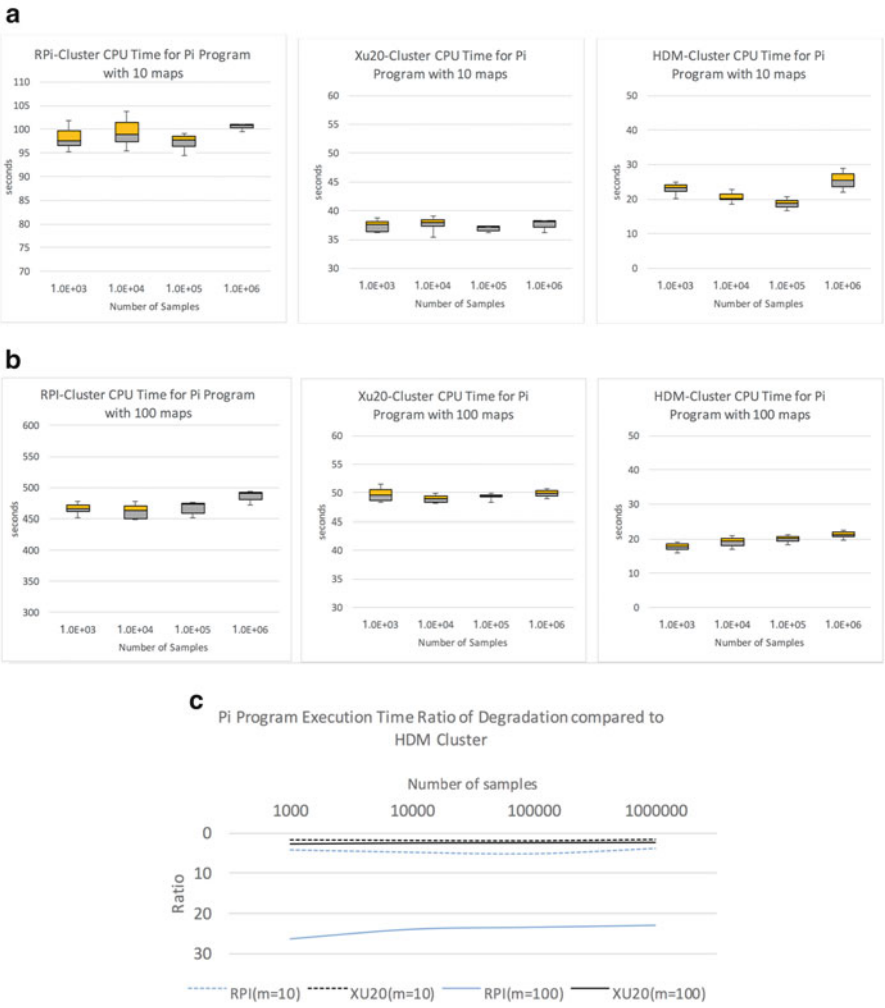


Fig. 15.9 CPU execution time versus number of *m* samples for computation of Pi benchmark in all clusters with (**a**) 10 maps (**b**) 100 maps (**c**) ratio of execution time for Rpi and Xu20 Cluster against HDM Cluster

from a text file as input and breaks it into words. It then emits a key/value pair of the word and a count value. Each reducer sums the count values for each word and emits a single key/value pair containing the word itself and the sum that word appears in the input files. For optimization, the reducer also imitates as a combiner on the map outputs to reduce the amount of data sent across the network by combining each word into a single record. In our experimentation, the Wordcount benchmark's goal is to observe the CPU bound workload of the three clusters.

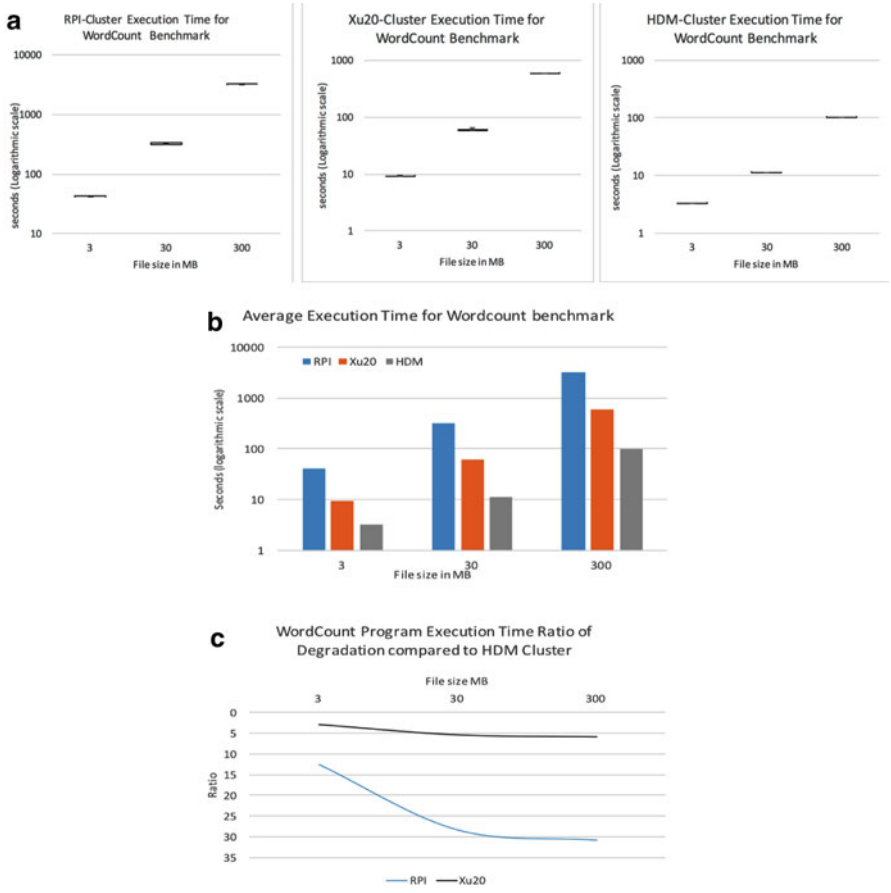


Fig. 15.10 (**a**) CPU execution time for the Wordcount benchmark for all clusters against input files sizes 3, 30, and 300 MB. (**b**) Average execution time for Wordcount on all clusters. (**c**) Ratio of performance degradation against HDM Cluster

In our experimentation, we generated three large files of sizes 3, 30, and 300 Megabytes, respectively. The Wordcount program was executed in the Hadoop environment for all the three clusters. Depending on the initial dataset size, Wordcount generates mappers for every HDFS container associated with the input files. For the datasets provided Wordcount generated a single mapper, four mappers, and 36 mappers, respectively. Each experiment was run on the clusters separately at least 10 times for statistical accuracy. Figure 15.10a shows the performance of CPU execution time, for the Wordcount benchmark for all clusters against input files sizes 3, 30, and 300 MB, in seconds on a logarithmic scale. Again, RPi Cluster performs four times worse (Fig. 15.10c) compared to Xu20 Cluster and 12.5 times worse compared to HDM Cluster due to the relatively slower processor clock speeds, slower memory read/write, and network latency. The effect of the slower clock speed

of the processor in the RPi nodes is clearly evident with smaller input file sizes of 3 MB. The average execution times of RPi and XU20 should be comparable since Wordcount generates only one mapper for each run resulting in a single container read by the mapper; however, the slower storage throughput with SD cards adds to the overall latency. With input file size 30 MB, Wordcount generates four mappers reading four containers from different nodes in the cluster, increasing the degree of parallelization thus reducing the overall CPU execution time.

Finally, with 300 MB as input file size, we observe execution time performance correlating with smaller datasets although the increased numbers of mappers should have improved the overall execution time. This is due to the fact that Wordcount generated 36 mappers for the job since there are only 19 nodes available (1 reserved for reducing job) in the Xu20 and RPi Clusters, the rest of the mappers would queue for the completion of previous mapper jobs resulting in increased overhead and reduced performance. Figure 15.10b shows the average CPU execution times for all three clusters with different input file sizes. Furthermore, we observe that the Wordcount program executing on Xu20 is 2.8 times slower compared to HDM Cluster for file size 3 MB. For larger file sizes, Xu20 is over five times slower compared to HDM Cluster. RPi Cluster, on the other hand, performs worse from 12 to 30 times slower compared to the HDM Cluster.

15.6 Discussion

In this chapter, we conducted an extensive study with varying parameters on the Hadoop cluster deployed using ARM-based single board computers. An overview of popular ARM-based SBCs Raspberry Pi, as well as HardKernel Odroid XU-4 SBCs, was presented. The work also detailed the capabilities of these devices and tested them using popular benchmarking approaches. Details on requirements, design, and architecture of clusters built using these SBCs were provided. Two SBC clusters based on RPi and XU-4 devices were constructed in addition to a PC-based cluster running in the virtual environment. Popular Hadoop benchmark programs such as Wordcount, TestDFSIO, and TeraSort were tested on these clusters and their performance results from the benchmarks were presented. This section presents a discussion of our findings and main lessons learned.

- Deployment of Clusters: Using low-cost SBCs is an amicable way of deploying a Hadoop cluster at a very affordable cost. The low-cost factor would encourage students to build their own clusters and to learn about installation, configuration, and operation of a cloud computing testbeds. The cluster also provides a platform for developers to build applications, test, and deploy in public/private cloud environments. The small size of the SBCs allows installation of up to 32 nodes in a single module for a 1 U rack mounting form factor. Further to this, these small clusters can be packaged for mobility and can be deployed in various emergency and disaster recovery scenarios.

- Hadoop configuration optimization: Section 15.4(a) comparison of CPU execution times using sysbench for both SBCs considered in this chapter. XU-4 devices in Xu20 Cluster perform better due to higher clock speeds and larger onboard RAM. Using sysbench we observed that increasing the number of cores in the CPU intensive benchmark, the execution time decreased. In Hadoop deployment configuration, we noticed that increasing the number of cores resulted in RPi Cluster to be irresponsive for heavier workloads. On the other hand, XU-4 boards performed well with an increased number of cores (up to 4). A possible explanation for this behavior is the Hadoop deployment setting where each core is assigned 852 MB of memory, additional cores running Hadoop tasks would have to request virtual memory from the slower SD cards resulting in poor performance leading to responsiveness. Although RPi devices are equipped with quad-core processors, due to the poor performing SD cards, it is inadvisable to use multiple cores for Hadoop deployment.

  In Hadoop deployment, not all of the available RAM onboard SBCs was utilized since we only allow one container to execute in YARN Daemon. The size of the container was set to 852 MB which is the maximum available onboard memory in a Raspberry Pi node. This was intentionally done in order to study the performance correlation with the similar amount of resources in both kinds of SBCs. In further experimentation, we notice that XU-4 devices are capable of handling up to four containers in each core at a time, resulting in better performance. We will further investigate the performance of all cores on the SBCs using Hadoop deployment of larger replication factors and a large number of YARN containers executing per node. On the HDM Cluster running Hadoop environment in a virtual machine, we note that higher replication factors resulted in a large number of errors due to replication overheads resulting in Hadoop stuck in an unrecoverable state. The SD cards are slow and the storage provided per node in the cluster is distributed over the network degrading the overall performance of the cluster. Raspberry Pi with slower network port at speeds 10/100 Mbps also poses a considerable degradation in network performance.

  On the other hand, Xu20 Cluster performed well comparatively with faster eMMC memory modules onboard the XU-4 devices. The SSD storage used in the HDM Cluster on the PCs provide the best performance in terms of storage IO although the network configuration of this cluster was a hindrance. We will consider using Network Attached Storage (NAS) attached to the master node where every rack would have a dedicated volume managed by Logical Volume Manager (LVM) that would be shared by all SBCs in the clusters.
- Power efficiency: A motivation for this study was to analyze the power consumption of SBC-based clusters. Due to their small form factor, SBC devices are inherently energy efficient, it is worth investigating if a cluster comprising of SBCs as nodes provides a better performance ratio in terms of power consumption and dollar cost. Although we did not measure the FLOPs per watt efficiency of either of our clusters, we notice wide inconsistencies in energy consumption results reported in the literature [17–22] for similar devices. This is due to the power measurement instruments varying results and inconsistencies in the design

of power supplies. RPi, as well as XU-4 devices, has no standard power supply and micro-USB-based power supply with unknown efficiency can be used. Since the total power consumed in the cluster is small, the efficiency of power supplies can make a big difference in overall power consumption. Nonetheless, WattsUp meters were effectively used to observe and analyze the power utilization for each task over the period of its execution in all experimentation.

It is difficult to monitor and normalize the energy consumption for every test run over a period of time. It was observed that the MapReduce jobs, in particular, tend to consume more energy initially while map tasks are created and distributed across the cluster, while a reduction in power consumption is observed towards the end of the job. For the computation of power consumption, we assumed max power utilization (stress mode) for each job, during a test run in the clusters. Based on the power consumption of each cluster and the dollar cost of maintaining the clusters (given in Table 15.4), a summary of average execution times, energy consumption, and cost of running various benchmark tasks is presented in Table 15.10.

15.7 Conclusions and Future Work

In this chapter, we investigated the Hadoop deployment on low-cost low-power ARM-based single board computers. We consider two kinds of popular platforms Raspberry Pi 2B and Odroid XU-4 using ARM Cortex Processors connected in a tree network topology. We perform various performance benchmarking tests on these two platforms testing performance metrics for CPU task execution times, removable memory modules, energy consumption, and network performance. We present the power consumption and estimate cost of power per year. Further to this, we configure and deploy Hadoop 2.6.2 on these clusters considering the limited capabilities of the SBCs. Various CPU-intensive and IO-intensive Hadoop benchmarks including computation of Pi using Monte Carlo method, Wordcount, TestDFSIO, and TeraSort were executed and performance results obtained. We carried out an in-depth analysis of energy consumption of these clusters and correlate performance with low-cost low-energy capabilities of these clusters.

Results from these studies show that while SBC-based clusters are energy efficient overall, the operation cost to performance ratio can vary based on the workload. In terms of power efficiency, for smaller workloads, the Xu20 Cluster outperforms the other clusters; however, with larger workloads, the Xu20 Cluster performance is comparable to HDM with the exception of TeraGen and TeraSort benchmarks. Similarly, in terms of dollar cost of operation for these clusters, the results heavily depend on execution time. For low-intensity workloads, the Xu20 Cluster outperforms the HDM Cluster; however, the TeraGen and TeraSort heavy workloads yield poor performance for Xu20 Cluster when compared to HDM

Table 15.10 Summary of execution time, energy consumption, and cost of running per job for all benchmarks

	CPU execution time (s)			Energy consumption (W) per job			Cost (Dollars) per job		
Test parameters	RPi Cluster	Xu20 Cluster	HDM Cluster	RPi Cluster	Xu20 Cluster	HDM Cluster	RPi Cluster	Xu20 Cluster	HDM Cluster
10 maps10^3 samples	98.47	37.37	22.86	1.27E + 0	8.17E − 1	1.26E + 0	6.35E − 2	4.08E − 5	6.28E − 5
10^4 samples	99.13	37.69	20.50	1.28E + 0	8.24E − 1	1.13E + 0	6.39E − 2	4.12E − 5	5.63E − 5
10^5 samples	97.90	36.97	18.92	1.26E + 0	8.08E − 1	1.04E + 0	6.31E − 2	4.04E − 5	5.20E − 5
10^6 samples	100.63	37.87	25.35	1.30E + 0	8.28E − 1	1.39E + 0	6.48E − 2	4.14E − 5	6.96E − 5
100 maps10^3 samples	465.68	49.62	17.84	6.00E + 0	1.08E + 0	9.80E − 1	3.00E − 1	5.42E − 5	4.90E − 5
10^4 samples	461.40	49.70	19.35	5.95E + 0	1.09E + 0	1.06E + 0	2.97E − 1	5.43E − 5	5.31E − 5
10^5 samples	470.26	49.43	20.12	6.06E + 0	1.08E + 0	1.10E + 0	3.03E − 1	5.40E − 5	5.52E − 5
10^6 samples	486.48	49.89	21.24	6.27E + 0	1.09E + 0	1.17E + 0	3.14E − 4	5.45E − 5	5.83E − 5
Filesize = 3 MB	41.12	9.25	3.29	5.30E − 1	2.02E − 1	1.81E − 1	2.65E − 5	1.01E − 5	9.03E − 6
30 MB	318.75	59.75	11.22	4.11E + 0	1.31E + 0	6.16E − 1	2.05E − 4	6.53E − 5	3.08E − 5
300 MB	3131.60	588.45	101.38	4.04E + 1	1.29E + 1	5.57E + 0	2.02E − 3	6.43E − 4	2.78E − 4

Cluster. The RPi Cluster consistently was outperformed by the other two clusters regardless of the variation in workloads.

For heavier workload application, such as big data applications, due to the inefficient performance of these devices, the SBC-based clusters may not be an appropriate choice. The overall cost of operation can be expensive mainly due to the inefficient onboard SBC resources resulting in larger execution times for job completion effectively ensuing increased operation costs. It is, however, possible to tweak Hadoop configuration parameters to adjust with given resources to improve the overall performance. At the moment, we intend to use these clusters for academic research and teaching. In the future, we will consider the use of NAS for RPi Cluster to improve the storage performance since the currently installed SD card storage provides a bottleneck. We will also study the effect of replication factor and containers per node in the Xu20 Cluster to tweak the performance on that cluster. Further, we intend to study newer SBC boards deployed in similar configurations with reliable power measurement and energy consumption analysis.

Acknowledgements This work is supported by the Robotics and Internet of Things Unit at the Research and Innovations Center at Prince Sultan University.

References

1. The Raspberry Foundation. https://www.raspberrypi.org (2015). Accessed online 20 Nov 2015
2. Chaari, R., Ellouze, F., Koubaa, A., Qureshi, B., Pereira, N., Youssef, H., Tovar, E.: Cyber-physical systems clouds: a survey. Comput. Netw. **18**, 260–278 (2016)
3. Gómez, A., Cuiñas, D., Catalá, P., Xin, L., Li, W., Conway, S., Lack, D.: Use of single board computers as smart sensors in the manufacturing industry. Proc. Eng. **132**, 153–159 (2015)
4. Grigoriev, S.N., Martinov, G.M.: An ARM-based multi-channel CNC solution for multi-tasking turning and milling machines. Proc. CIRP. **46**, 525–528 (2016)
5. Fernandes, S.L., Bala, G.J.: ODROID XU4 based implementation of decision level fusion approach for matching computer-generated sketches. J. Comput. Sci. **16**, 217–224 (2016)
6. Grisenthwaite, R.: ARMv8-A Technology Preview. Online. Accessed 31 Oct 2011
7. Kiepert, J.: Creating a Raspberry Pi-Based Beowulf Cluster. Boise State University, pp. 1–17 (2013). http://coen.boisestate.edu/ece/files/2013/05/Creating.a.Raspberry.Pi-Based.Beowulf.Cluster_v2.pdf
8. Abrahamsson, P., Helmer, S., Phaphoom, N., Nicolodi, L., Preda, N., Miori, L., Angriman, M., Rikkila, J., Wang, X., Hamily, K., Bugoloni, S.: Affordable and energy-efficient cloud computing clusters: the Bolzano raspberry pi cloud cluster experiment. In: 2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom), vol. 2, pp. 170–175 (2013)
9. Cox, S.J., Cox, J.T., Boardman, R.P., Johnston, S.J., Scott, M., O'Brien, N.S.: Iridis-pi: a low-cost, compact demonstration cluster. Clust. Comput. **17**(2), 349–358 (2014)
10. Tso, F.P., White, D.R., Jouet, S., Singer, J., Pezaros, D.P.: The Glasgow raspberry pi cloud: a scale model for cloud computing infrastructures. In: 2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops (ICDCSW), pp. 108–112. IEEE, Philadelphia, PA (2013)
11. Whitehorn, J.: Raspberry flavored Hadoop. On-Line access: http://www.idatasci.com/uploads/1/4/6/6/14661274/jamiewhitehorn_raspberryflavouredhadoop_annotated.pdf

12. Expedient Data Center Cost Estimator. https://www.expedient.com/data-center-build-vs-buy-calculator/
13. Toor, S., Osmani, L., Eerola, P., Kraemer, O., Lindén, T., Tarkoma, S., White, J.: A scalable infrastructure for CMS data analysis based on OpenStack Cloud and Gluster file system. J. Phys. Conf. Ser. **513**, 062047 (2014). https://doi.org/10.1088/1742-6596/513/6/062047
14. Baun, C.: Mobile clusters of single board computers: an option for providing resources to student projects and researchers. SpringerPlus. **5**, 360 (2016). PMC. Web. 16 Aug 2016
15. Huang, S., Huang, J., Dai, J., Xie, T., Huang, B.: The HiBench benchmark suite: characterization of the MapReduce-based data analysis. In: International Conference on Data Engineering Workshops (ICDEW), March 2010
16. Ivanov, T., Niemann, R., Izberovic, S., Rosselli, M., Tolle, K., Zicari, R.V.: Performance evaluation of enterprise big data platforms with HiBench. In: IEEE Trustcom/BigDataSE/ISPA, Helsinki (2015)
17. Ge, R., Feng, X., Song, S., Chang, H.C., Li, D., Cameron, K.W.: PowerPack: energy profiling and analysis of high-performance systems and applications. IEEE Trans. Parallel Distrib. Syst. **21**(5), 658–671 (2010)
18. Cloutier, M.F., Paradis, C., Weaver, V.M.: Design and analysis of a 32-bit embedded high-performance cluster optimized for energy and performance. In: Hardware-Software Co-Design for High-Performance Computing (Co-HPC), 2014, New Orleans, LA (2014)
19. Psaroudakis, I., Kissinger, T., Porobic, D., Ilsche, T., Liarou, E., Tözün, P., Ailamaki, A., Lehner, W.: Dynamic fine-grained scheduling for energy-efficient main-memory queries. In: The Tenth International Workshop on Data Management on New Hardware (DaMoN'14) (2014)
20. Schöne, R., Treibig, J., Dolz, M.F., et al.: Tools and methods for measuring and tuning the energy efficiency of HPC systems. J. Sci. Program. **22**(4), 273–283 (2014)
21. Piga, L., Bergamaschi, R.A., Breternitz, M., Rigo, S.: Adaptive global power optimization for Web servers. J. Supercomput. **68**, 1088 (2014)
22. Ilsche, T., Hackenberg, D., Graul, S., Schöne, R., Schuchart, J.: Power measurements for compute nodes: improving sampling rates, granularity and accuracy. In: Sixth International Green Computing Conference and Sustainable Computing Conference (IGSC), Las Vegas, NV (2015)
23. Hackenberg, D., Ilsche, T., Schöne, R., Molka, D., Schmidt, M., Nagel, W.E.: Power measurement techniques on standard compute nodes: a quantitative comparison. In: 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Austin, TX, pp. 194–204 (2013)
24. Divakaran, D., Le, T., Gurusamy, M.: An online integrated resource allocator for guaranteed performance in data centers. IEEE Trans. Parallel Distrib. Syst. **25**(6), 1382–1392 (2014)
25. Dalvandi, A., Gurusamy, M., Chua, K.C.: Time-Aware VMFlow Placement, Routing, and Migration for Power Efficiency in Data Centers. IEEE Trans. Netw. Serv. Manag. **12**(3), 349–362 (2015)
26. Cook, G.: How clean is your cloud? Catalysing an energy revolution. Greenpeace International, Amsterdam (2012)
27. Data Center Efficiency Assessment: Scaling up energy efficiency across the data center industry. Natural Resources Defense Council, August 2014. https://www.nrdc.org/sites/default/files/data-center-efficiency-assessment-IP.pdf
28. Xiao, Z., Song, W., Chen, Q.: Dynamic resource allocation using virtual machines for cloud computing environment. IEEE Trans. Parallel Distrib. Syst. **24**(6), 1107–1117 (2013)
29. Qureshi, B., Koubaa, A.: Power efficiency of a SBC based Hadoop cluster. In: Proceedings of EAI Smart Societies, Infrastructure, Technologies and Applications (SCITA 2017), Jeddah, Saudi Arabia, pp. 52–60, 27–29 Nov 2017
30. Conejero, J., Rana, O., Burnap, P., Morgan, J., Caminero, B., Carrión, C.: Analyzing Hadoop power consumption and impact on application QoS. Futur. Gener. Comput. Syst. **55**, 213–223 (2016)

Chapter 16
Parallel Iterative Solution of Large Sparse Linear Equation Systems on the Intel MIC Architecture

Hana Alyahya, Rashid Mehmood, and Iyad Katib

16.1 Introduction

Finding a solution to sparse linear equation systems is at the core of scientific computing. Numerous scientific, engineering, and smart city applications require the solution of sparse linear systems [1–5].

The general form of the sparse linear system is $Ax = b$, where A is a sparse matrix, x is the solution vector, and b is a dense vector. There are two well-known categories of the numerical methods for solving linear equations of the form $Ax = b$, namely direct and iterative methods. Direct methods are robust but more expensive in terms of memory usage because the data structures used to store the matrix in this method need to be updated frequently while the algorithm is executed [6]. On the other hand, iterative methods start with an initial guess and modify the approximation solution on each iteration until it converges. Although iterative methods do not guarantee convergence, they have better performance than direct methods in terms of speed. The iterative methods can be further classified into stationary iterative methods such

H. Alyahya (✉)
Information Technology Section, Institute of Public Administration in Makkah Al-Mukaramah Region Women Branch, Jeddah, Saudi Arabia
e-mail: alyahyah@ipa.edu.sa

R. Mehmood
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: RMehmood@kau.edu.sa

I. Katib
Computer Science Department, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: iakatib@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_16

as Jacobi and Gauss-Seidel (GS) [7, 8] and non-stationary iterative methods such as the Conjugate Gradient (CG).

The Sparse Matrix-Vector multiplication (SpMV) operation is an important part of many iterative solvers of linear equation systems, both stationary (e.g. the Jacobi method) and non-stationary (e.g. the Conjugate Gradient (CG) method) [9]. The Jacobi iterative method consists of many iterations of the SpMV operation. However, SpMV is considered a bottleneck due to its intensive computational and storage needs. There is a huge volume of literature available on iterative solvers. For example, see [10–14]. A survey on the existing iterative methods used to solve linear equation systems is presented in [15].

Sparse matrices that arise from real-life problems are typically large, but consist of a relatively small number of non-zero elements. Efficient storage formats are required to store only the non-zero elements, such that the memory usage is minimised, while providing flexible and fast access to the matrix non-zero elements. Many sparse storage formats have been proposed over the years, the best known of these include the Coordinate format (COO), the Compressed Sparse Row (CSR) format, the Modified Sparse Row (MSR), the Blocked Compact MSR format, and the Diagonal format among others [16–19].

16.1.1 Motivation and Problem Statement

The designs of the current computing systems bring new challenges and opportunities. Compared to the earlier systems, the contemporary systems show that computing performance gets better with the increasing number of cores [20]. The multi-core and many core devices, and increasing storage capabilities allow developers to optimise their algorithms and benefit from those technologies. The Intel Many Integrated Core (MIC) architectures consist of a highly parallel engine and efficient processor architecture that achieves a high performance through the utilisation of a large number of cores, like vector register and high bandwidth on package memory. The first generation of the Intel MIC architectures was Intel Knights Corner (KNC). The second generation of Intel Xeon Phi was based on the Intel Knights Landing (KNL) chip [21]; and these devices could be used as a stand-alone processor as well as a coprocessor. Knights Hill (cancelled in 2017) and Knights Mill were announced subsequently with increasing focus on machine learning and deep learning applications. Many applications are being ported to the MIC devices because of the compatibility of the MIC architectures with CPUs and their programming simplicity (see e.g. [22]).

Many researches have focused on designing an efficient solver for sparse linear systems of equations because of its importance and its usage in a large number of scientific and engineering applications. Although the method itself is very important, a greater awareness in terms of hardware is needed to better take advantage of some of the new features available in today's systems to gain improved computational performance. There is limited work on optimising the current iterative solvers, or designing new algorithms and implementations that will benefit from Intel Xeon Phi coprocessor capabilities. A number of issues

need to be considered when implementing iterative methods on modern many core architectures including the Intel MIC architecture. As large sparse matrices with diverse sparsity patterns must be dealt with, the storage format used to store the sparse matrix can affect the performance. In addition to the storage scheme, iterative solvers should also take the advantage of Single Instruction Multiple Data (SIMD) registers and the many cores available on Intel MIC for good performance. Therefore, these issues require selecting the best storage scheme for a given sparse matrix and efficiently implementing the Jacobi iterative method on Intel MIC architecture.

In this chapter, we investigate the performance of parallel implementations of the Jacobi method on the Knights Corner (KNC) architecture. We implement Jacobi with two storage formats, Compressed Sparse Row (CSR) and Modified Sparse Row (MSR), and measure their performance in terms of execution time, offloading time, and speedup. We report results of sparse matrices with over 28 million rows and 640 million non-zero elements acquired from 13 diverse application domains. The experimental results show that our Jacobi parallel implementation on MIC achieves speedups of up to 27.75× compared to the sequential implementation. It also delivers a speedup of up to 3.81× compared to a powerful node comprising 24 cores in two Intel Xeon E5-2695v2 processors.

This is an extended version of our earlier work [23]. The earlier work reported results of parallel implementation of SpMV on MIC while in this chapter we report the results of implementing parallel Jacobi method on both multi-cores and many core architectures.

The rest of the chapter is organised as follows: Sect. 16.2 presents the background material on the solution of large sparse linear systems, sparse matrix storage formats, and sparse matrix-vector multiplication. A basic background on Intel MIC architecture is also provided. Section 16.3 reviews the literature on iterative methods, sparse storage formats, and SpMV on Intel MIC architecture. A discussion on the challenges in MIC implementation of iterative methods, SpMV and sparse matrix storage formats is provided. The gaps in the current literature have been identified. Section 16.4 discusses our methodology to efficiently implement Jacobi on Intel MIC and the algorithms that have been proposed to improve the performance. In Sect. 16.5, we analyse and compare the performance of our implementations with the sequential implementation and with the performance of Multi-Cores. Section 16.6 concludes the chapter and provides future research directions.

16.2 Background

16.2.1 Solving Large Sparse Linear Equation Systems

There are two classes of solvers for solving sparse linear systems of the form $Ax = b$, namely direct solvers and iterative solvers. Direct solvers have a finite set of

procedures that give an exact solution. They are robust but difficult to parallelise and consume memory. In contrast, iterative solvers have a sequence of approximation solutions, starting with an initial guess and improving the solution until it converges to something very close to an exact solution. Although iterative methods do not guarantee convergence, they are scalable, amenable to parallelism, and do not consume memory. In this section, the most common direct and iterative solvers used to solve sparse linear systems are presented.

Direct Methods

Direct methods as stated above have a finite set of procedures to achieve an exact solution. They are also robust and predictable. However, as the size of the matrix increases, they become insufficient due to fill-in during the factorisation phase. In the next sections, two of the well-known direct methods, the Gaussian Elimination and LU Factorisation, will be briefly described.

Gaussian Elimination

Gaussian elimination is an efficient direct method used to solve linear equation systems. The augmented matrix for the system is first written and then reduced to echelon form using elementary row operations. Finally, the matrix is solved using back substitution. Since Gaussian elimination alters the matrix however, it is difficult to use for solving large sparse linear systems.

LU Factorisation/Decomposition

LU Factorisation is one of the direct methods for solving linear equation systems. It forms an important part of many computer algorithms. LU decomposition is based on factorising the coefficient matrix A into the multiplication of lower and upper triangular matrices. The coefficient matrix A becomes:

$$A = LU \tag{16.1}$$

where L is the lower triangular of the coefficient matrix A and U is the upper triangular. After the factorisation process, one back substitution and one forward substitution is performed in order to solve the system $Ax = b$.

Iterative Methods

As discussed earlier, iterative methods generate a sequence of approximation solutions starting with an initial solution and improving this solution in each

iteration until it converges to the exact solution. Iterative methods can be classified into stationary iterative methods such as Jacobi, Gauss-Seidel, and Successive Over-Relaxation (SOR) and non-stationary iterative methods like Krylov Subspace methods. The next sections provide more details about these four iterative methods.

Jacobi Method

The Jacobi method is one of the simplest stationary iterative methods for determining the solution of the system of linear equations. Jacobi starts with an initial guess for the unknowns x and obtains new results in each iteration until the values of x approach the exact solution. For each iteration, the essential computation of the Jacobi method is as follows:

$$x_i^{(k)} = a_{ii}^{-1} \left(b_i - \sum\nolimits_{j \neq i} a_{ij} x_j^{(k-1)} \right) \tag{16.2}$$

for $0 \leq i < n$, where a_{ij} represents the values in row i and column j of matrix A, a_{ii} denotes the diagonal elements assuming that none of them is zero, x_i^k is the i-th element in the k-th iteration. When implementing Jacobi, we need two vectors. One to store the previous value of x and the other to store the new value of x. In each iteration, the new x vector is updated with the previous x. This makes Jacobi amenable to parallelism. However, it may take a long time to converge.

Gauss-Seidel Method

The Gauss-Seidel method is an improved method of the Jacobi iterative solver [24]. The Gauss-Seidel method uses the most recent value of approximation of solution x which is superior to Jacobi because it converges quickly. In addition, Gauss-Seidel does not require much storage because there is a single iteration vector updated in each iteration. However, the Gauss-Seidel method is completely non-parallelisable due to its use of the most recent approximation solution.

Successive Over-Relaxation (SOR) Method

Successive Over-Relaxation (SOR) is an iterative method for solving linear equation systems. It is based on the Gauss-Seidel method but it moves more quickly towards a solution. By introducing a new parameter, namely relaxation factor ω, the SOR iteration becomes:

$$x_i^{(k+1)} = x_i^{(k)} + \omega \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij} x_j^{(k)} \right), i = 1, 2, \ldots, n. \tag{16.3}$$

The good choices of the relaxation factor ω are between [0, 2]. If the relaxation factor is greater than 0 and less than 1, it is termed under-relaxed and if it is greater than 1, it is termed over-relaxed and if it is equal to 1, then it is reduced to Gauss-Seidel. SOR is used to improve any iterative methods that are slow to converge but it depends on the choice of the optimal relaxation factor.

Krylov Subspace Methods

Krylov Subspace Methods is one of the non-stationary iterative method for solving systems of linear equations [25]. They are usually used with large matrices to find a suitable approximation in a shorter amount of time. The computation in Krylov Subspace Methods is based on the matrix vector multiplication and the independent updated vector. Krylov Subspace Methods converge faster than stationary methods, but they are difficult to apply in some matrices that require multiple iteration vectors [9]. The common Krylov Subspace Methods are the Conjugate Gradient (CG) for symmetric and positive definite matrices and the Generalised Minimal Residual Method (GMRES) for general matrices.

16.2.2 Test of Convergence for Iterative Methods

For iterative methods, it is necessary to test the convergence of the method in each iteration. The iterative methods should be stopped when the stopping criteria is met. The most common stopping criteria is:

$$\left\| x^{k+1} - x^k \right\| \leq \varepsilon \tag{16.4}$$

where ε is a predetermined threshold. The convergence of the Jacobi iterative method depends on the properties of the sparse matrix A and sometimes the choice of the initial guess may reduce the number of iterations to converge.

16.2.3 Sparse Matrix Storage Formats

Sparse matrix storage formats are used to store only the non-zero elements and their locations, which results in saving memory and improving performance.

Several sparse storage formats have been proposed over the years. In this section, the common storage formats are reviewed: the Coordinate format (COO), the Compressed Sparse Row (CSR) format, the Modified Sparse Row (MSR), the ELLPACK (ELL) format, and the Diagonal format (DIA).

Coordinate Storage (COO)

The coordinate storage format (COO) is one of the simplest formats, using three arrays to store the sparse matrix. The first, namely val, stores the non-zero elements of the sparse matrix arbitrarily. The other two arrays store the column and row indices of the non-zero elements [26].

Compressed Sparse Row (CSR)

The compressed row storage format CSR is a solution for efficiently storing the sparse matrices and reducing the memory overhead. CSR stores the sparse matrices as follows: it has three arrays, val[nnz] array of size nnz, where nnz is the number of non-zero in matrix A. The val[nnz] array is used to store the value of the non-zero elements. Col_in[nnz] is an array of size nnz and it stores the column indices of the non-zero elements. Row_ptr[$m + 1$] is an array of size $m + 1$ and it stores the non-zero elements in each row [27].

Modified Sparse Row (MSR)

The Modified Sparse Row storage format (MSR) is a modified version of CSR. It works in a similar way to CSR, except that the diagonal elements are stored in a separate array. MSR does not need to store the column indices of the diagonal elements and this makes it more efficient than CSR [19]. Iterative methods such as Jacobi can store the diagonal entries as 1 upon the diagonal elements to reduce the amount of computations by replacing division with multiplication.

The ELLPACK (ELL) Format

The ELLPACK format has two arrays, val and col_in of size $n \times k$, where k is the maximum number of non-zero entries in each row [26]. The val array stores the non-zero elements and col_in stores the column indices of the non-zero entries. The val array is padded with zero if the rows contain less than k non-zero elements. The ELLPACK format becomes insufficient however if all the rows have low entries, except one row that has large non-zero entries.

Diagonal Storage (DIA)

Diagonal Storage (DIA) is a special format for storing diagonally structured sparse matrices. It is similar to the ELLPACK format but it is more restricted and compact. DIA stores values according to their diagonal and ignores the column indices which reduce the memory bandwidth. However, storing the zeros of the diagonal can waste memory.

16.2.4 Sparse Matrix-Vector Multiplication (SpMV)

The sparse matrix-vector multiplication kernel is shown in Eq. (16.5), where A is a square sparse matrix $N \times N$, and x and y are vectors of length N. The matrix A is multiplied by vector x and added to vector y

$$y = y + Ax \tag{16.5}$$

Due to the irregular pattern of the non-zero values in the sparse matrix A, SpMV is considered to be one of the most time-consuming kernels. As a result, its performance is poor.

16.2.5 Intel MIC Architecture

The Intel Many Integrated Core Architecture was developed by Intel. The key feature of this architecture is that there are many Intel® processor cores in one chip. Another advantage is that it supports many programming languages such as standard C, Fortran, and C++. The flexibility of compiling and running code in any of the Intel® Xeon® processors is also an important feature. In addition, it supports the most widely used parallel programming models such as OpenMP and MPI [28]. The Intel Xeon Phi coprocessor is based on Intel MIC architecture. It supports up to 61 small x86 cores that work together. It has 8 memory controllers and supports up to 16 GDDR channels. It has a transfer speed of 5.5GT/s (in theory) and a level 2 cache memory. The instruction level cache has a size of 32 KB and the data cache has a size of 32 KB [29]. Xeon Phi has two execution modes: offload execution and native (coprocessor) execution [30]. In the offload mode, the host send part of the code to Xeon Phi and the output data is sent back from the coprocessor to Xeon. Whereas, in the native mode, the code is run natively in the coprocessor. Figures 16.1 and 16.2 shows the two modes.

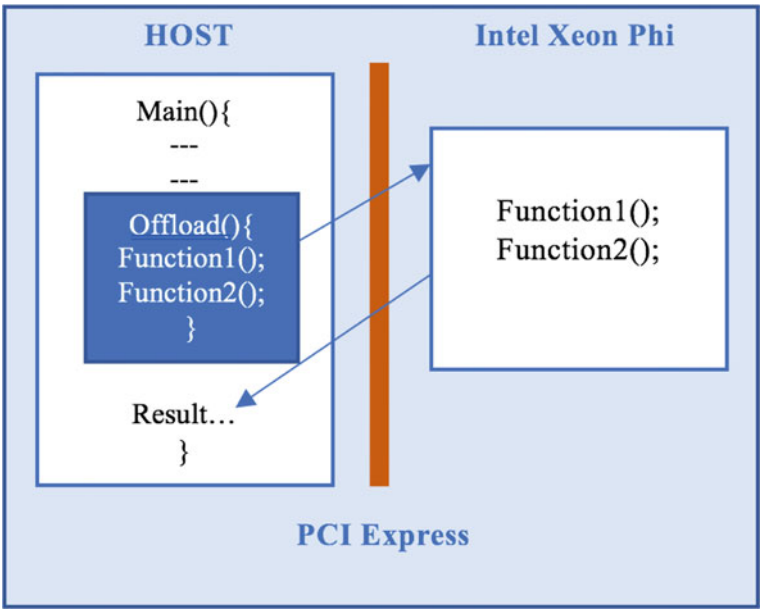
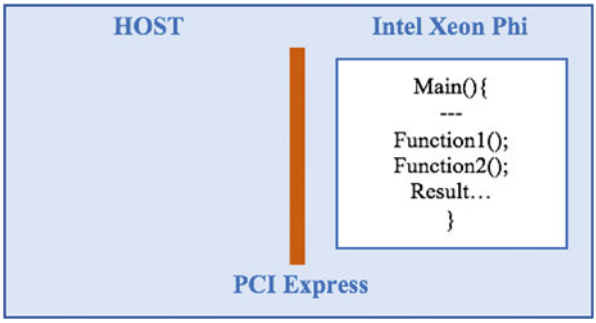


Fig. 16.1 Offloading mode in Xeon Phi

Fig. 16.2 Native mode in Xeon Phi



16.3 Related Work

This section presents a survey of the work related to this chapter. The parallel techniques used for solving linear equation systems on modern architectures are reviewed. The main focus will be on Intel Many Integrated Core (MIC) architecture, which is a highly parallel engine and an efficient processor architecture that achieves a high performance through the utilisation of a large of number of cores, like the vector register and a high bandwidth on package memory.

16.3.1 Parallel Methods for Solving Linear Equation Systems

In this section, the main methods used for solving linear equation systems are reviewed. This includes only the parallel techniques as the serial ones are not in the scope of this chapter. The focus will be the methods employed, the architecture used for implementation, the data structure used for storing the matrices, the size, and properties of the matrices and the results.

Mehmood and Crowcroft [19] focus on the steady-state analysis of Continuous Time Markov Chains (CTMCs), which is used for performance analysis in many computer and communication applications. They used a blocked version of the Compact MSR scheme [9, 31] to store the CTMCs matrices and the parallel Jacobi iterative method for steady-state solutions for large CTMCs. They used a 24-node processor bank, three CTMCs case studies, the PRISM tool for generating the model of the case studies and C language, with MPICH implementation. They report the solution of sparse systems with over a billion states and 18 billion non-zero elements.

Tang et al. [32] implemented the Preconditioned Conjugate Gradient (PCG) on Intel MIC architecture. They used the compressed sparse row (CSR) format to store the sparse matrices. Some matrices from the University of Florida's Sparse Matrix Collection (UFSPARSE) were used for testing. The experimental results showed that while the number of non-zero elements increases the speedup, the execution time decreases.

Li et al. [33] evaluated the Conjugant Gradient (CG) iterative solver used for solving sparse linear systems. They considered large-scale power systems applications. They implemented a serial version of Conjugant Gradient Normal Residual (CGNR) and CGNR with Jacobi preconditioning; and a parallel version of CGNR with Jacobi preconditioning and CG with Jacobi preconditioning. The implementation was based on multi-core CPUs and many core GPUs. The results show that implementing CG on a GPU gives a better performance than a CPU, considering that the matrices are large and well- conditioned.

Yan et al. [34] implemented a serial and parallel iterative Jacobi method for solving sparse linear systems on a CPU and a GPU. They used the CSR format to store the sparse matrix. The parallel version was implemented on a hybrid multi-core system containing a general CPU and GPU. They improved the performance of Jacobi in terms of data storage and the access mode of CUDA. The results showed that the parallel version of Jacobi implemented on a GPU has a better performance than a CPU and proved that the optimisation scheme was effective and feasible.

16.3.2 Sparse Matrix-Vector Multiplication

Numerous studies have been conducted on the SpMV computation, as it is used in many scientific and engineering applications. Ye et al. [35] reported an implemen-

tation of SpMV computations on Intel MIC architecture using OpenMP, MPI, and hybrid MPI/OpenMP models. Their study showed that the hybrid model performed well on Intel MIC architecture.

Maeda and Takahashi [30] implemented SpMV on CPU, MIC, and GPU clusters and evaluated the performance of each cluster. They showed that MIC outperformed other accelerators using a small number of MPI processes. However, the performance decreased when the number of MPI process increased, due to communication overhead. Saule et al. [36] studied the performance of the Intel Xeon Phi coprocessor for SpMV and focused on the memory bandwidth. Their results showed that Xeon Phi could not reach its peak performance due to the memory latency and not the bandwidth.

Maeda and Takahashi [37] evaluated the performance of parallel Sparse Matrix-Vector Multiplication (SpMV) on different architectures such as CPU, Intel MIC, and GPU clusters. They used the CSR storage format to store the sparse matrices. The results showed that the performance of parallel SpMV using a CPU cluster increased by 42.57 in comparison to the single process. In some matrices, the performance was low due to load imbalance and communication overheads. The performance of parallel SpMV on the accelerators was higher on the CPU clusters in the matrices with a large amount of non-zero, or when using a small number of MPI processes. However, when the number of MPI processes became large, the performance of the parallel SpMV on MIC was low, due to communication overhead. To overcome this, Maeda and Takahashi proposed to apply the Segmented Scan (SS) method to the MIC cluster to improve the parallel SpMV. As a result, the performance of the imbalanced matrices with 64 MPI processes was increased.

Ahamed and Magoules [38] analysed and evaluated the performance of Sparse Matrix-Vector Multiplication (SpMV) and Krylov methods on GPUs. They considered different methods for solving sparse linear systems with symmetric and non-symmetric matrices. They applied different storage formats and showed their impact on the performance of the iterative solvers.

16.3.3 Studies Related to the Intel Xeon Phi Coprocessor

This section focuses on studies of the performance of the Intel Xeon Phi coprocessor. The Intel Xeon Phi coprocessor has many cores that can handle many threads with large vector units. It is a great choice for developing high performance applications that have a computation of more than a trillion times. Many research papers have studied the behaviour of Intel Xeon Phi in terms of its software and hardware.

Some of the work focuses on implementing existing methods on coprocessors. For example, Cramer et al. [28] used OpenMP style programming to test the efficiency of Intel Xeon Phi when running standard applications. They analysed the performance of Intel Xeon Phi by using simple benchmarks and by implementing sparse CG kernels. Cramer et al. used Native Execution and Language Extensions

for Offload (LEO) to program the coprocessor and investigated the Intel MIC architecture suitability using the Roofline model. They showed the performance of Intel Xeon Phi compared to the performance of a large SMP production system.

Estebanez et al. [39] performed several experiments to test the performance of Thread-Level speculation (TLS) on the Intel Xeon Phi coprocessor. The authors used three real applications as a benchmark against a synthetic one. The results of their experiments showed that the scalability of the Intel Xeon Phi coprocessor achieved up to 240 GB/s whereas the AMD Opteron 6376 achieved up to 51.2 GB/s. However, it required higher execution times compared to the traditional shared memory because the applications have an irregular nature when used for performing TLS techniques.

Other research focused on improving the performance of Intel Xeon Phi. Dongarra et al. [40] improved LAPCK algorithms to make them work efficiently on Intel Xeon Phi. They designed API that allows the developer to access the low level of the coprocessor's architecture through abstract means. They tried to achieve maximum parallelism when using Intel Xeon Phi coprocessors by defining a new method which split the algorithms into several tasks and scheduled them using the directed acyclic graph (DAG). To test their suggested methods, they used Intel Xeon Phi represented as a group of servers and workstations. As a comparison, they used a multi-core system with two sockets and 8 core Intel Xeon E5-2670 (Sandy Bridge) processors. They implemented three of the linear algebra solvers: QR, Cholesky, and LU factorisation. The results showed that their implementation had a better performance than implementing the MKL libraries on the CPU. The final study is by Chen et al. [41], in which the authors used the OpenACC standard to automatically translate the source code to the Intel offload code. They used two well-known kernels, namely the matrix multiplication and the Jacobi iterative method for testing. They were implemented on CPUs, Intel MIC, and GPUs. Two optimisation techniques were used, namely communication and SIMD. The performance evaluation showed that the two kernels implemented on Intel MIC had a better speedup than the GPUs and CPUs.

16.4 Methodology

This section discusses the implementation of SpMV on MIC as it forms an important part of the Jacobi method. Several versions of the Jacobi iterative method will also be proposed and implemented to efficiently utilise the features of Intel MIC architecture. Sparse matrices were collected from the University of Florida online matrix collection [42]. Only square matrices were collected and other matrices ignored. SpMV is based on off-diagonal matrices only while Jacobi is based on both the off- and on-diagonal non-zero elements. It should be noted that in the Jacobi method, only matrices that had non-zero elements in the main diagonal were taken.

The sparse matrices are from different application domains including the following: optimisation problem, directed graph, undirected random graph, circuit simu-

Table 16.1 Applications name and abbreviation

Application name	Abbreviation
Optimisation problem	OP
Directed graph	DG
Undirected random graph	URG
2D/3D problem	2D/3D P
Circuit simulation problem	CSP
Undirected graph	UG
Directed weighted graph	DWG
Undirected multigraph	UMG
Computational fluid dynamics problem	CFDP
Structural problem	SP
Electromagnetics problem	EMP
Model reduction problem	MRP

lation problem, undirected graph, directed weighted graph, undirected multigraph, computational fluid dynamics problems, structural problem, and electromagnetics problem. Table 16.1 shows the applications and their abbreviations. For simplicity, the abbreviations will be used for the remainder of the chapter.

There are total of 39 real sparse matrices in the Matrix Market format. The details of the matrices are given in Table 16.2 including the dimensions, the non-zero elements, the non-zero elements per row, and the application domain. Figures 16.3, 16.4, 16.5, and 16.6 plot the sparsity structures of some matrices from the collection.

The matrices were converted to CSR and MSR before applying SpMV and Jacobi computations. The parallelisation process works as follows. The instruction is divided among the threads. Each thread will complete the calculation and bring the results back. This process will continue until the loop is finished. The numbers of threads used are: 1, 4, 16, 32, 64, 128, 236, and 240. The following sections discuss the implementation of SpMV and Jacobi on Intel MIC in more detail.

16.4.1 SpMV

SpMV is essential to the Jacobi iterative method for solving linear equation systems. For this reason, SpMV is first implemented on MIC before Jacobi to see how the parallelisation works. Figure 16.7 shows the pseudocode of the parallel SpMV. In line 2, an OpenMP pragma is added to the outer loop, so each thread will carry out the computation separately until they reach the end of the loop. The outer loop will begin with zero until it reaches the size of Matrix A, which is n in this case. The inner loop will start from the first row and end at the last row that contains non-zero elements. Line 6 shows the main operation. Each row will be multiplied by the values of vector x and will then be added to vector y. When the outer loop finishes, the result will be returned and sent back to the CPU. Figure 16.8 shows

Table 16.2 Sparse matrices properties

Name	Size	nnz	nnz/row	Application domain
nlpkkt240	28.0M	401.2M	14.33	OP
arabic-2005	22.7M	640.0M	28.14	DG
rgg_n_2_24_s0	16.8M	132.6M	7.90	URG
delaunay_n24	16.8M	50.3M	3.00	UG
nlpkkt200	16.2M	232.2M	14.30	OP
wb-edu	9.8M	57.2M	5.81	DG
nlpkkt160	8.3M	118.9M	14.25	OP
indochina-2004	7.4M	194.1M	26.18	DG
circuit5M	5.6M	59.5M	10.71	CSP
ljournal-2008	5.4M	79.0M	14.73	DG
cage15	5.2M	99.2M	19.24	DWG
soc-LiveJournal1	4.8M	69.0M	14.23	DG
channel-500x100x100-b050	4.8M	42.7M	8.89	UG
kron_g500-logn21	2.1M	91.0M	43.41	UMG
HV15R	2.0M	283.1M	140.33	CFDP
wikipedia-20051105	1.6M	19.8M	12.08	DG
G3_circuit	1.6M	4.6M	2.92	CSP
Flan_1565	1.6M	59.5M	38.01	SP
af_shell10	1.5M	27.1M	17.96	SP
cage14	1.5M	27.1M	18.02	DWG
Hook_1498	1.5M	31.2M	20.83	SP
Atmosmodl	1.5M	10.3M	6.93	CFDP
StocF-1465	1.5M	11.2M	7.67	CFDP
Geo_1438	1.4M	32.3M	22.46	SP
Serena	1.4M	33.0M	23.69	SP
in-2004	1.4M	16.9M	12.23	DG
Atmosmodd	1.3M	8.8M	6.94	CFDP
dielFilterV2real	1.2M	24.8M	21.47	EMP
hollywood-2009	1.1M	57.5M	50.46	UG
dielFilterV3real	1.1M	45.2M	40.99	EMP
bone010	986.7K	36.3M	36.82	MRP
Ldoor	952.2K	23.7M	24.93	SP
audikw_1	943.7K	39.3M	41.64	SP
Emilia_923	923.1K	21.0M	22.71	SP
Fault_639	638.8K	14.6M	22.9	SP
inline_1	503.7K	18.7M	37.05	SP
RM07R	381.7K	37.5M	98.16	CFDP
Pwtk	217.9K	5.9M	27.19	SP
nd24k	72.0K	14.4M	199.91	2D/3D P

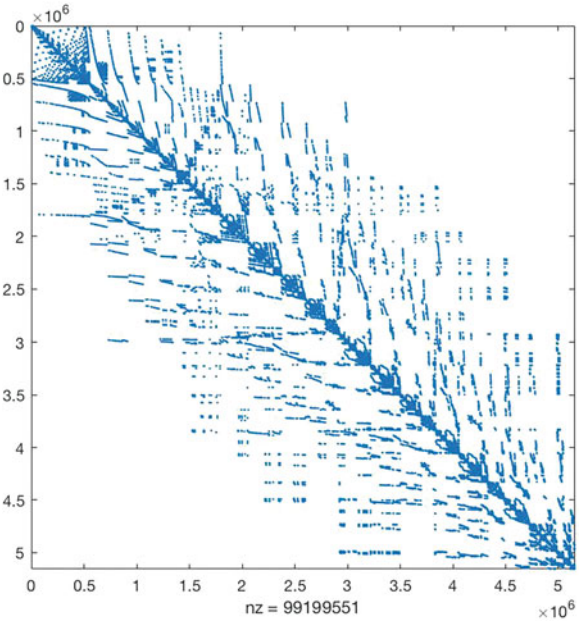


Fig. 16.3 Sparsity of matrix cage15

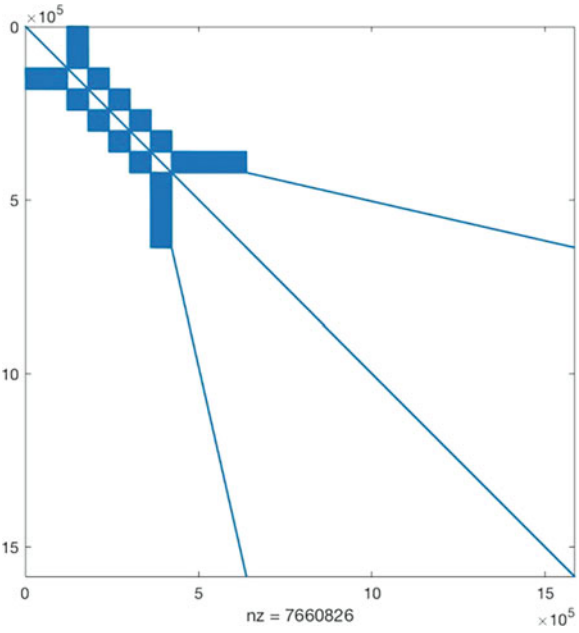


Fig. 16.4 Sparsity of matrix g3_circuit

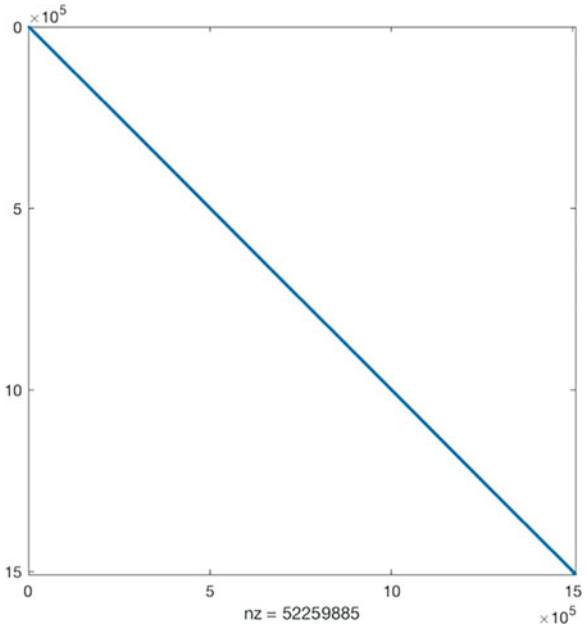


Fig. 16.5 Sparsity of matrix af_shell10

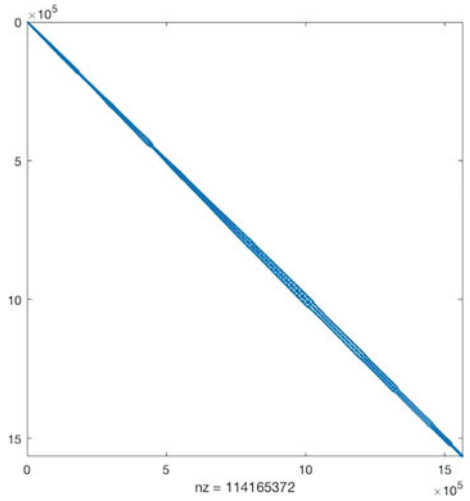


Fig. 16.6 Sparsity of matrix flan_1565

the parallelisation process of SpMV computation, where y_1, y_2, and y_n represent the y vector, coloured boxes represent the sparse matrix non-zero elements, x_1, x_2, and x_n represent the x vector and thread_0, thread_1, and thread_n represent the thread numbers. As shown in the figure, each thread multiplies a row with the whole vector x. At the end, the summation of the y vector is performed.

Algorithm 1 Parallel Sparse Matrix Vector Multiplication With CSR, y=y+Ax

```
1: procedure SpMV-CSR(val, x, n, row_ptr, col_in)
2:     #pragma omp parallel for private(j)
3:     for i = 0 : n do
4:         y = 0.0
5:         for j = row_ptr[i] : row_ptr[i + 1] do
6:             y+ = val[j] * x[col_in[j]]
7:         end for
8:     end for
9: end procedure
```

Fig. 16.7 Parallel sparse matrix-vector multiplication with CSR, $y = y + Ax$

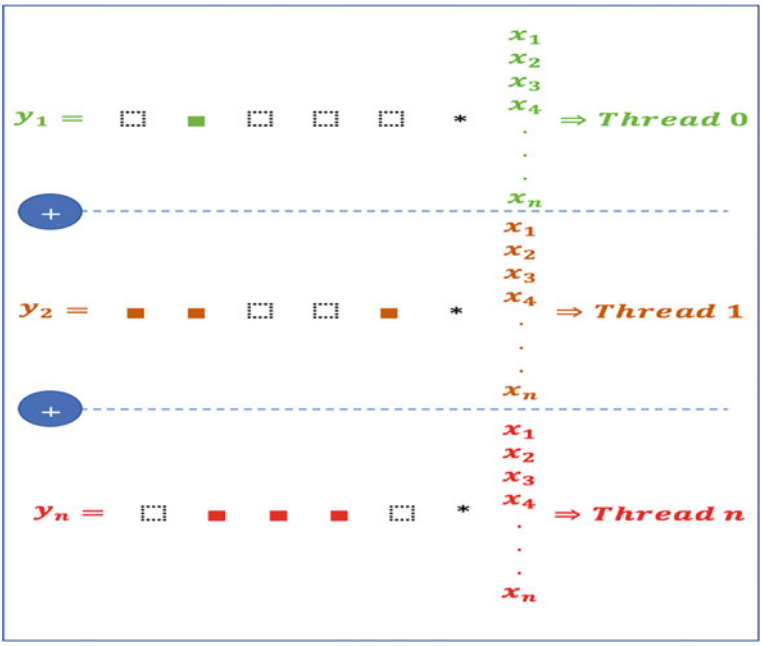


Fig. 16.8 SpMV parallelisation

16.4.2 Jacobi

After implementing SpMV on MIC, the Jacobi method is considered. Four versions of the Jacobi iterative method were implemented in addition to the standard method. They were first implemented on a CPU and then offloaded onto MIC. Three steps were followed. Firstly, the sparse matrix was read in CSR and MSR format, since the downloaded matrices are in the Matrix Market (MM) format. This is done using CPU as it has a large memory compared to MIC. Secondly, when the matrix and vector were ready, the part of the code that has the Jacobi computation was offloaded to MIC. Then, OpenMP pragmas were used to parallelise the "for" loops. Finally, the results were sent from the coprocessor to the host; and the host prints the results and the execution time in seconds.

Figure 16.9 shows the serial version of Jacobi, with single storage for full matrix CSR. As discussed earlier, the CSR has three vectors val, col_in, and row_ptr. In order to implement Jacobi, in addition to these vectors, the following are needed: the size of the matrix (n), (b) vector, two vectors (x) and (x_new) to hold the values of the approximation solutions and two variables (a_ii) to save the non-zero elements in the main diagonal and (sum) to save the summation of SpMV. The method continues with the "while" loop until the conditions are satisfied. There are two conditions. The first is the Distance, which is when the approximation solutions converge to the actual solution. The second condition is when the iteration reaches the maximum iteration defined by user. Inside the "while" loop, there are two nested "for" loops that contain the main operations of the Jacobi method. The outer loop will begin with zero until it reaches the size of Matrix A, which is n in this case. The inner loop will start from the first row and end at the last row that contains non-zero elements. In the inner loop, we have an "if" condition to check if the non-zero elements are in the diagonal or the off-diagonal. In the case of the off-diagonal, the non-zero values will be multiplied with the whole vector x and the results will be added to the variable "sum". If they are in the diagonal, then they are assigned to the "a_ii" variable. The "if" condition is followed by the next operation, which is dividing the subtraction of vector b with sum by the diagonal values. Finally, the value of x is updated by the new value in line 18.

Figure 16.10 shows the parallel Jacobi iterative method with single storage for full matrix (CSR). In this algorithm, the first modification is made, namely parallelising the "for" loops. It is the same process as before, except that in line 4 and 18, an openMP pragma is added to both the outer loop and the "for" loop for updating the value of x.

Algorithm 2 Serial Jacobi Iterative Method with Single Storage for Full Matrix (CSR))

```
1:  procedure JACOBI-SERIAL(n, val, col_in, row_ptr, b, x, x_new)
2:      while (Distance(x, x_new, n) >= epsilon && iter_counter < max_iter) do
3:
4:          for i = 0 : n do
5:              double a_ii;
6:              double sum = 0.0;
7:              for j = row_ptr[i] : row_ptr[i + 1] do
8:                  if (col_in[j]! = i) then
9:                      sum+ = val[j] * x[col_in[j]]
10:                 else
11:                     a_ii = val[j];
12:                 end if
13:             end for
14:             x_new[i] = (b[i] − sum)/a_ii;
15:         end for
16:
17:         for i = 0 : n do
18:             swap(&x[i], &x_new[i])
19:         end for
20:
21:     end while
22: end procedure
```

Fig. 16.9 Serial Jacobi iterative method with single storage for full matrix (CSR)

Algorithm 3 Parallel Jacobi Iterative Method (CSR: Single Matrix Storage)

```
1: procedure JACOBI-VERSION1(n, val, col_in, row_ptr, b, x, x_new)
2:     while (Distance(x, x_new, n) >= epsilon && iter_counter < max_iter) do
3:
4:         #pragma omp parallel for private(j)
5:         for i = 0 : n do
6:             double a_ii;
7:             double sum = 0.0;
8:             for j = row_ptr[i] : row_ptr[i + 1] do
9:                 if (col_in[j]! = i) then
10:                    sum+ = val[j] * x[col_in[j]]
11:                else
12:                    a_ii = val[j];
13:                end if
14:            end for
15:            x_new[i] = (b[i] − sum)/a_ii;
16:        end for
17:
18:        #pragma omp parallel for
19:        for i = 0 : n do
20:            swap(&x[i], &x_new[i])
21:        end for
22:
23:    end while
24: end procedure
```

Fig. 16.10 Parallel Jacobi iterative method (CSR: Single Matrix Storage)

Algorithm 4 Parallel Jacobi Iterative Method (MSR: Separate Diagonal Storage)

```
1: procedure JACOBI-VERSION2(n, OffDiagonal, Diagonal, col_in, row_ptr, b, x, x_new)
2:     while (Distance(x, x_new, n) >= epsilon && iter_counter < max_iter) do
3:
4:         #pragma omp parallel for private(j)
5:         for i = 0 : n do
6:             double sum = 0.0;
7:             for j = row_ptr[i] : row_ptr[i + 1] do
8:                 sum+ = OffDiagonal[j] * x[col_in[j]]
9:             end for
10:            x_new[i] = (b[i] − sum)/Diagonal[i];
11:        end for
12:
13:        #pragma omp parallel for
14:        for i = 0 : n do
15:            swap(&x[i], &x_new[i])
16:        end for
17:
18:    end while
19: end procedure
```

Fig. 16.11 Parallel Jacobi iterative method (MSR: Separate Diagonal Storage)

Figure 16.11 shows the code for the second modification which stores the sparse matrix using separate diagonal storage MSR. In this algorithm, the "if" condition is removed as the diagonal values are stored in a separate array, therefore there is no

Algorithm 5 Parallel Jacobi Iterative Method (Version 3)

```
1:  procedure JACOBI-VERSION3(n, OffDiagonal, OneOverDiagonal, col_in, row_ptr, b, x, x_new)
2:      while (Distance(x, x_new, n) >= epsilon && iter_counter < max_iter) do
3:
4:          #pragma omp parallel for private(j)
5:          for i = 0 : n do
6:              double sum = 0.0
7:              for j = row_ptr[i] : row_ptr[i + 1] do
8:                  sum+ = OffDiagonal[j] * x[col_in[j]]
9:              end for
10:             x_new[i] = (b[i] − sum) * OneOverDiagonal[i];
11:         end for
12:
13:         #pragma omp parallel for
14:         for i = 0 : n do
15:             swap(&x[i], &x_new[i])
16:         end for
17:
18:     end while
19: end procedure
```

Fig. 16.12 Parallel Jacobi iterative method (Version 3)

need to check if the non-zero elements are in the main diagonal or otherwise. The main diagonal entries are stored in an array called (diagonal) and the off-diagonal entries stored in another array called (OffDiagonal). In line 8, the off-diagonal non-zero elements are multiplied by vector x and added to the variable sum. Then in line 10, the same was done as in the previous algorithms, the division of subtracting b vector with sum by the diagonal elements is stored in the new vector of x. Finally, the value of x is updated with x_new.

Figure 16.12 shows the code after storing the diagonal values as 1 upon diagonal in order to avoid division in Jacobi. In line 10, dividing the subtraction of vector b from the sum variable is replaced by multiplication. The remaining parts remain the same as in the previous algorithms.

Finally, Fig. 16.13 shows the last modification, which is parallelising the distance function using openMP pragma. An openMP pragma is added to the Distance function to parallelise the "for" loop. The remaining parts of the algorithm are not different to the previous algorithms.

16.5 Performance Evaluation

This section presents the experimental results of implementing four versions of Jacobi iterative methods on Multi-Cores and Intel MIC. The system configuration and setup are first described in Sect. 16.5.1. In Sect. 16.5.2, the experimental results for Multi-Cores using different versions of Jacobi are shown. The experimental results for implementing different versions of Jacobi on Intel MIC are shown in

Algorithm 6 Parallel Jacobi Iterative Method with Parallel Distance Function

```
1: procedure JACOBI-VERSION4(n, OffDiagonal, OneOverDiagonal, col_in, row_ptr, b, x, x_new)
2:     while (Parallel(Distance(x, x_new, n)) >= epsilon && iter_counter < max_iter) do
3:
4:         #pragma omp parallel for private(j)
5:         for i = 0 : n do
6:             double sum = 0.0
7:             for j = row_ptr[i] : row_ptr[i + 1] do
8:                 sum+ = OffDiagonal[j] * x[col_in[j]]
9:             end for
10:            x_new[i] = (b[i] − sum) * OneOverDiagonal[i];
11:        end for
12:
13:        #pragma omp parallel for
14:        for i = 0 : n do
15:            swap(&x[i], &x_new[i])
16:        end for
17:
18:    end while
19: end procedure
```

Fig. 16.13 Parallel Jacobi iterative method with parallel distance function

Table 16.3 Experimental environments

	Multi-Cores	Intel MIC
Model name	Two Intel(R) Xeon(R) CPU E5-2695 v2 (Ivy Bridge EP)	Intel Phi 5110P Coprocessor (Knights Corner)
Clock speed	2.40 GHz	1.05 GHz
No. of cores	12 cores each (up to 24 threads)	60 (up to 240 threads)
Cache	30 MB SmartCache	30 MB L2
Memory	DDR3 64 GB	GDDR5 8 GB

Sect. 16.5.3. Finally, a comparison between the performance of Multi-Cores and Intel MIC is presented in Sect. 16.5.4.

16.5.1 Experimental Setup

In order to evaluate the effectiveness and efficiency of the algorithms, several numerical experiments were conducted on real large sparse linear equation systems taken from the University of Florida Sparse Matrix Collection. The Aziz supercomputer was used for the experiments. It is a high performance computer located in King Abdulaziz University, Jeddah. It is one of the top 500 supercomputers in the world and one of the top 10 supercomputers in the Kingdom of Saudi Arabia [43]. The offloading mode was used for executing Jacobi because the native mode is not supported on Aziz. Multi-Cores and Intel MIC were used to implement the Jacobi versions. Table 16.3 shows the experimental environments.

Algorithm 7 Calculating Offloading & Execution Time

```
1: double tStart = omp_get_wtime(); //Start Offload timing
2: #pragma offload target (mic) in(..) inout(..)
3: {
4:       double tStart_exec = omp_get_wtime(); //Start Execution timing
5:       Jacobi()
6:       double tEnd_exec = omp_get_wtime(); //End Execution timing
7: }
8: double tEnd =omp_get_wtime();//End Offload timing
9: Offloading_time=tEnd−tStart;
10: Execution_time=tEnd_exec−tStart_exec;
11: print(Execution_time,Offloading_time);
```

Fig. 16.14 Calculating execution time and offloading time

The focus was in the performance of the Jacobi method on both systems in terms of execution time in seconds, offloading time in seconds and speedup. The execution time and offloading time were calculated using different numbers of threads: 1, 4, 16, 32, 64, 128, and 236. It should be noted that the execution time is the time taken to execute Jacobi in seconds and the offloading time is the time taken to offload Jacobi to MIC in seconds and this includes the execution time. Figure 16.14 shows how the execution time and the offloading time are calculated.

The speedup can be calculated as:

$$s_P = \frac{T_s}{T_p} \tag{16.6}$$

where T_s is the execution time of sequential implementation and T_p is the execution time of parallel implementation. The code is run 5 times and the average execution time taken, excluding the first run (warm-up). It is assumed that the right-hand side vector (b) is equal to zero if the matrix does not have the right-hand side vector. The approximation solution vector (x) was first set to be 1 upon n.

16.5.2 Experimental Results of Jacobi on Multi-Core Nodes

In this section, the performance evaluation of the Jacobi iterative method on Multi-Cores is described. Four versions of Jacobi were implemented in addition to the standard method. The average execution time per iteration was calculated in seconds for all versions and the best execution time for each version was plotted. Figure 16.15 shows a comparison between the best execution times on all four versions. Note that all the four versions appear to have similar performance with some variations. We have done a more detailed analysis of the four versions for both SpMV and Jacobi iterative method on MIC and multi-core and have found Version 4 to have overall better performance. We have therefore used Version 4 in all the experiments reported hereon. Figure 16.16 shows the average execution time per

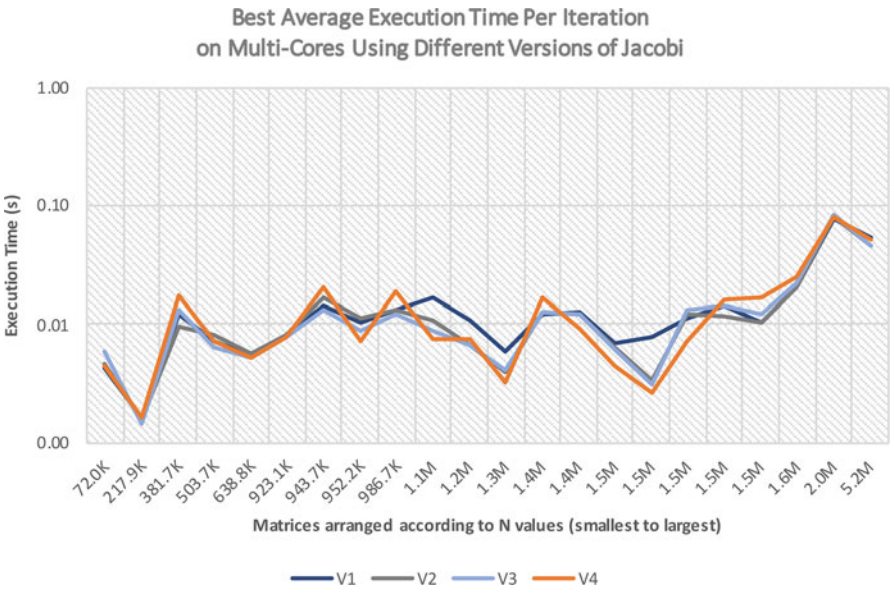


Fig. 16.15 Comparison between different versions of Jacobi on multi-cores

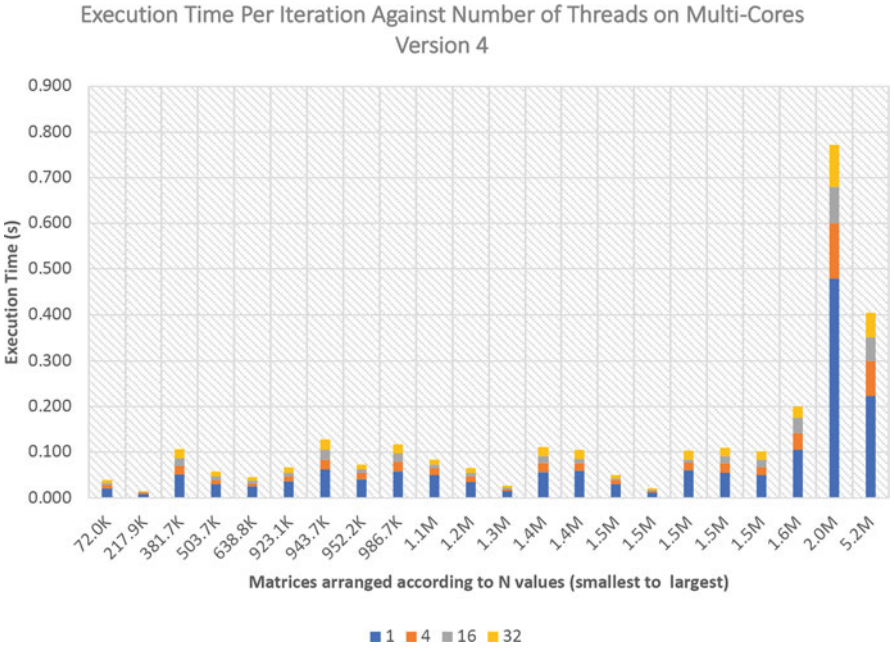


Fig. 16.16 Execution time against number of threads on the multi-core node (v4)

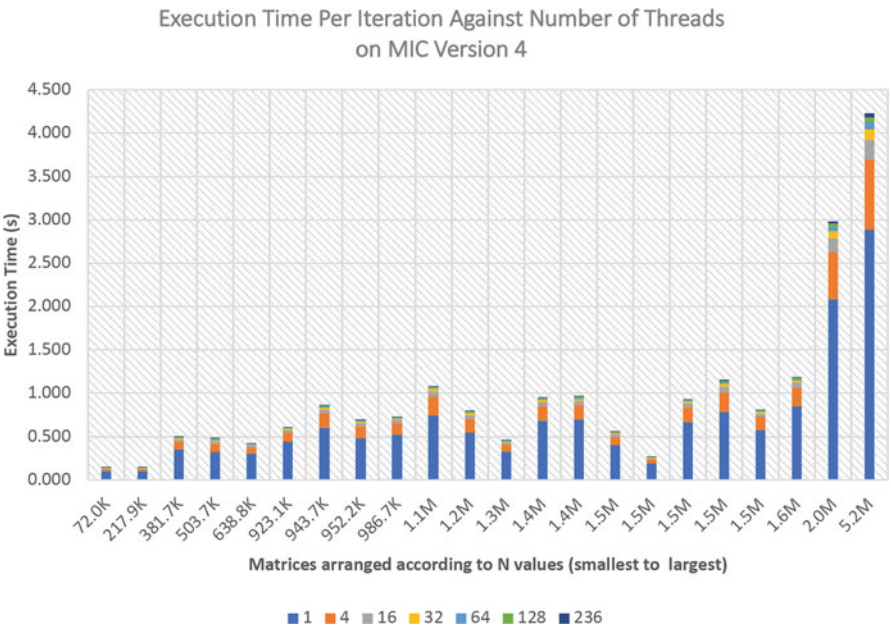


Fig. 16.17 Execution time against number of threads on MIC

iteration for version 4. For all matrices, the best execution time per iteration is found when the number of threads is 16.

16.5.3 Experimental Results of Jacobi on Intel MIC

In this section, the performance evaluation of the Jacobi iterative method is described. As discussed earlier, there are four versions of Jacobi, excluding the standard method. The average execution time and offloading time per iteration was calculated in seconds for all versions. The best execution time in each version was taken and compared. The results of the last version, Version 4, are shown in Fig. 16.17. For all matrices, the best execution time per iteration is found when the number of threads is 236. It has better performance than third version by up to 2.32.

Figure 16.18 shows the best average execution time per iteration using different versions of Jacobi according to the non-zero elements per row from smallest to largest. It can clearly be seen that all versions have the same behaviour when the number of non-zero per row increases.

Figure 16.19 shows the offloading time against the number of threads for Version 4. The matrices are arranged according to size from smallest to largest. It is observed

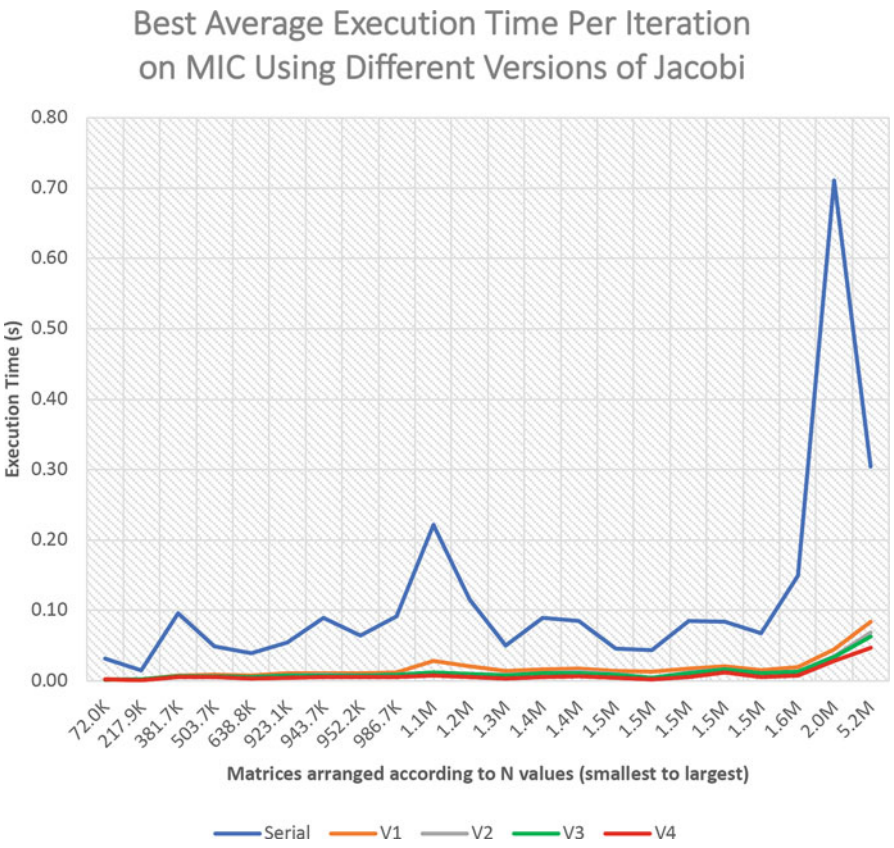


Fig. 16.18 Best average execution time per iteration on MIC

from Fig. 16.19 that for all the matrices, the best offloading time occurs when the number of threads is equal to 236.

16.5.4 Comparison Between a Multi-Core Node and Intel MIC

In this section, a comparison between the performance of Multi-Cores and Intel MIC is shown in terms of execution time and speedup. Figure 16.20 shows the best average execution time per iteration on Intel MIC and Multi-Cores using Version 4. The matrices are arranged according to size, from smallest to largest. The results show that Intel MIC has a better execution time than Multi-Cores. Figure 16.21 shows the maximum speedup achieved by implementing Version 4 of Jacobi on Multi-Cores and Intel MIC compared to the sequential execution using matrices of different sizes. The maximum speedup achieved by Intel MIC is 27.5 while in Multi-

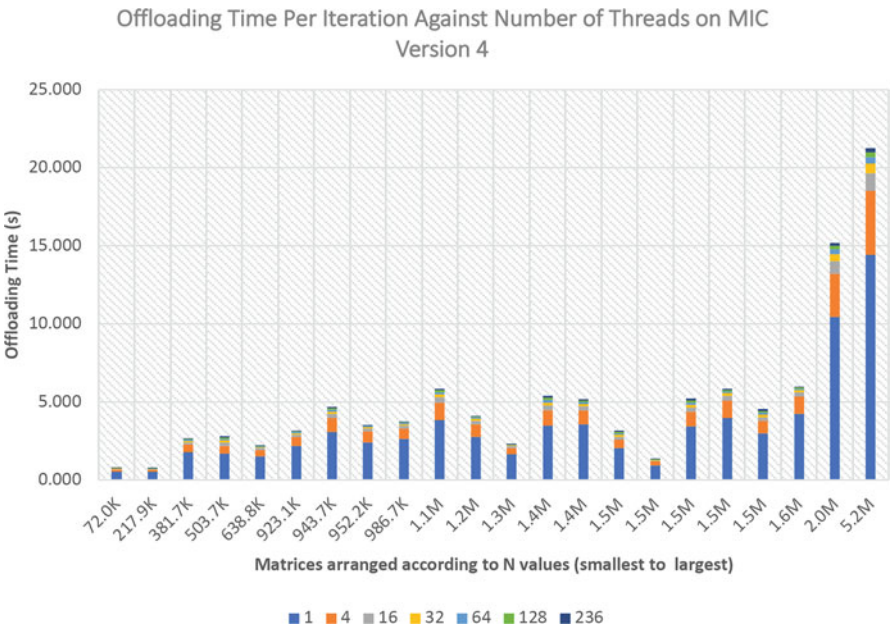


Fig. 16.19 Offloading time per iteration against number of threads on MIC

Cores it is 16.45 compared to the sequential execution. Both maximum speedups are found in the matrix of size 1.1 million. The performance of Jacobi in Intel MIC is 3.81× faster than the performance of Jacobi in Multi-Cores.

16.5.5 Sparse Matrix-Vector Multiplication (SpMV) Results

We have mentioned earlier in Sect. 16.1 that the Sparse Matrix-Vector multiplication (SpMV) operation is an important part of many iterative solvers of linear equation systems including the Jacobi iterative method. To implement SpMV efficiently on Intel MIC architecture, we have followed three steps. Firstly, we read the sparse matrix in CSR format since the downloaded matrices are in the Matrix Market (MM) format. This is done using the CPU as it is having large memory compared to MIC. Secondly, when the matrix and vector is ready, we offloaded the part of the code that has the SpMV computation to MIC. After the SpMV offloaded to MIC, we use OpenMP pragmas to parallelise the "for" loops. Finally, the results are sent from the coprocessor to the host and the host prints the results and the execution time. We calculate the execution time and offloading time using different number of MIC cores (or threads) 1, 4, 8, 16, 32, 64, 128, and 240. In addition, we calculate amount of memory used by each matrix. Note that the execution time is the time taken to execute the SpMV computation and offloading time is the time taken to offload the

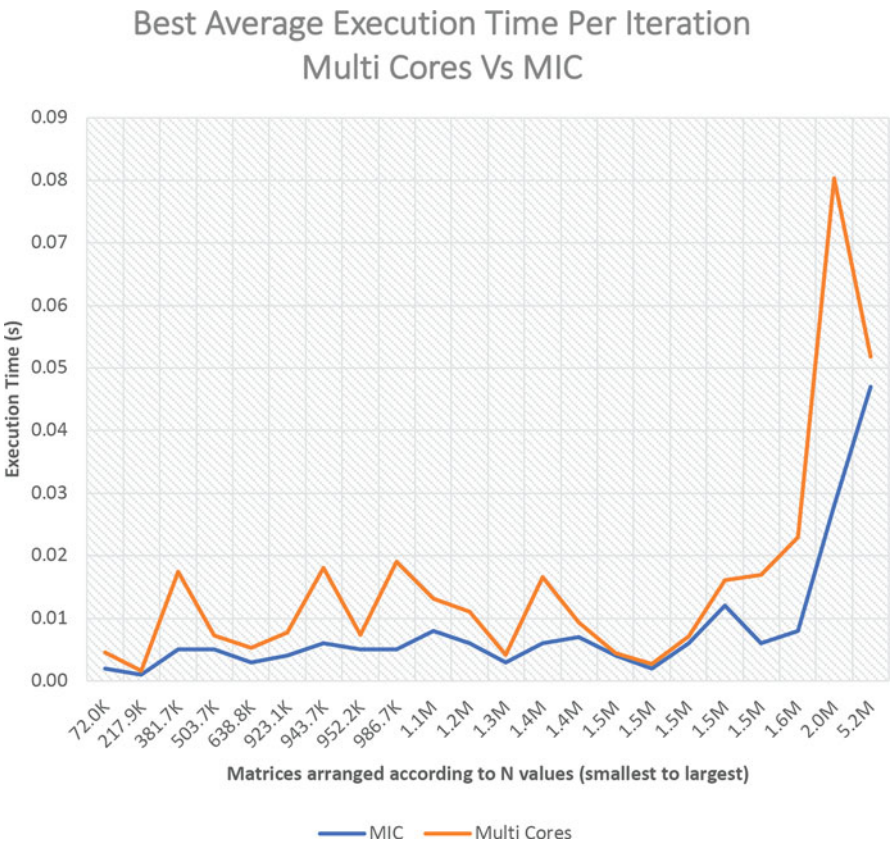


Fig. 16.20 Best average execution time per iteration multi-cores vs Intel MIC

SpMV computation to MIC and that includes the execution time. For simplicity, we divided the matrices into four groups according to their sizes, Groups 1, 2, 3, and 4, each group has eight matrices. The details of these matrices have been given earlier in Sect. 16.4 (see Table 16.2).

Figure 16.22 shows the execution time against the number of threads for the eight (largest) matrices in Group 1. It can be seen that using 240 threads gives the best execution time compared to the other number of cores. On average, the execution time of parallel implementation with 240 threads is 4.59× faster than the serial one. Further details about the SpMV performance can be found in our earlier work [23], where the main focus was on the SpMV computations.

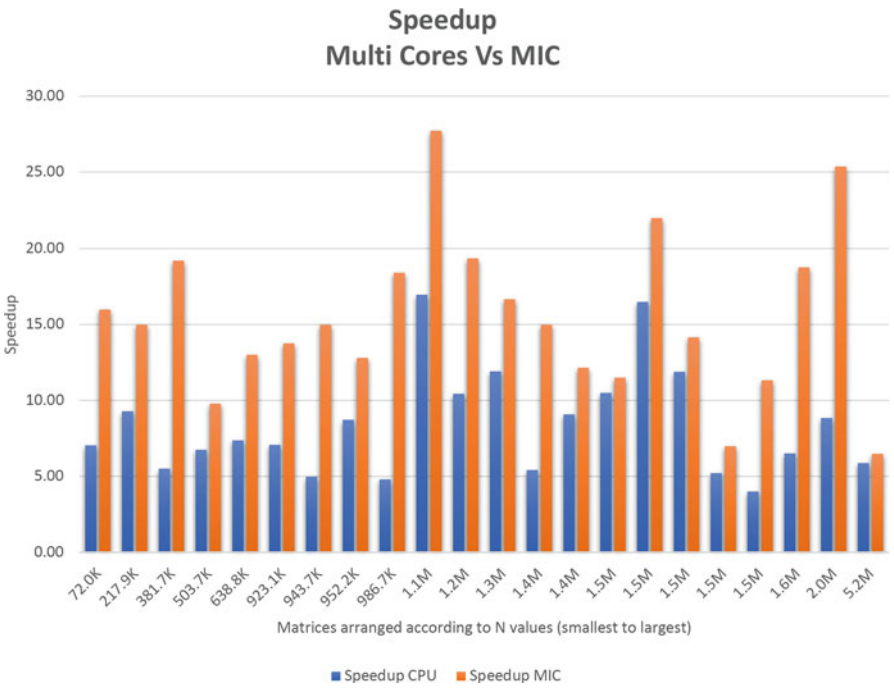


Fig. 16.21 Intel MIC vs multi-cores speedups

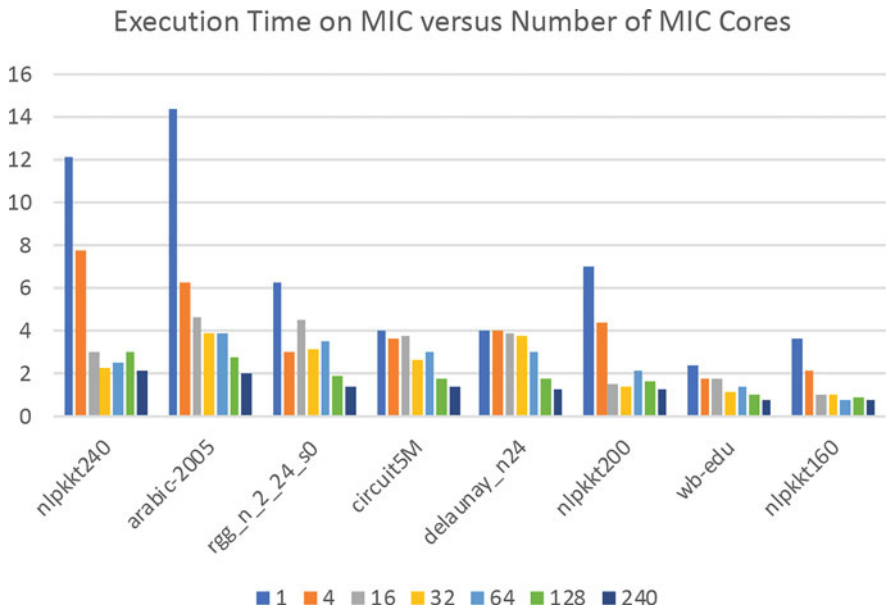


Fig. 16.22 Execution time on MIC against the number of MIC Cores

16.6 Conclusion and Future Work

The main focus of this chapter was to efficiently implement the Jacobi method on Intel MIC architecture. A performance analysis and evaluation were provided of the parallel Jacobi iterative method on the first generation of Intel Xeon Phi coprocessor and Intel MIC, named Knights Corner (KNC). Four versions of Jacobi were implemented in addition to the sequential implementation. Jacobi was implemented with two storage formats: Compressed Sparse Row (CSR) and Modified Sparse Row (MSR). The offload programming model was used to offload the Jacobi computations. OpenMP was used to do the parallel implementation on Intel MIC. The performance was measured in terms of the execution time, offloading time, and speedup. Results of the sparse systems were reported with over 28 million equations and 640 million non-zero elements. The experimental results show that the performance of this implementation achieves speedups of up to 27.75× compared to sequential implementation. MIC has a speedup of up to 3.81× compared to the Multi-Cores.

Future work will look into further analysis of the parallel Jacobi method for sparse linear equations systems of larger sizes from diverse application domains with the aim to further improve the performance. The implementation of the Jacobi iterative method would be attempted using a new scheme to efficiently store the sparse matrices.

Acknowledgments The experiments reported in this chapter were performed on the Aziz supercomputer at King Abdulaziz University, Jeddah, Saudi Arabia.

References

1. Mehmood, R., Alturki, R., Zeadally, S.: Multimedia applications over metropolitan area networks (MANs). J. Netw. Comput. Appl. **34**, 1518–1529 (2011)
2. Mehmood, R., Meriton, R., Graham, G., Hennelly, P., Kumar, M.: Exploring the influence of big data on city transport operations: a Markovian approach. Int. J. Oper. Prod. Manag. **37**, 75–104 (2017)
3. Mehmood, R., Graham, G.: Big data logistics: a health-care transport capacity sharing model. Proc. Comput. Sci. **64**, 1107–1114 (2015)
4. Mehmood, R., Lu, J.A.: Computational Markovian analysis of large systems. J. Manuf. Technol. Manag. **22**, 804–817 (2011)
5. Altowaijri, S., Mehmood, R., Williams, J.: A quantitative model of grid systems performance in healthcare organisations. In: 2010 Int. Conf. Intell. Syst. Model. Simul., pp. 431–436 (2010)
6. Saad, Y.: Iterative methods for sparse linear systems. Society for Industrial and Applied Mathematics (2003)
7. Golub, G.H., Van Loan, C.F.: Matrix Computations (2013)
8. Ford, W.: Chapter 20: Basic iterative methods. In: Ford, W. (ed.) Numerical Linear Algebra with Applications, pp. 469–490. Academic, Boston (2015)
9. Mehmood, R.: Disk-based techniques for efficient solution of large Markov chains. PhD Thesis, School of Computer Science, University of Birmingham (2004)

10. Kwiatkowska, M., Mehmood, R., Norman, G., Parker, D.: A symbolic out-of-core solution method for Markov models. Electron. Notes Theor. Comput. Sci. **68**, 589–604 (2002)
11. Kwiatkowska, M., Parker, D., Yi Zhang, Y., Mehmood, R.: Dual-processor parallelisation of symbolic probabilistic model checking. In: The IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004. (MASCOTS 2004). Proceedings, pp. 123–130. IEEE (2004)
12. Mehmood, R., Crowcroft, J., Elmirghani, J.M.H.: A parallel implicit method for the steady-state solution of CTMCs. In: 14th IEEE International Symposium on Modeling, Analysis, and Simulation, pp. 293–302. IEEE (2006)
13. Mehmood, R.: A survey of out-of-core analysis techniques in stochastic modelling. Technical Report CSR-03-7, School of Computer Science, University of Birmingham, Birningham (2003)
14. Mehmood, R., Parker, D., Kwiatkowska, M.: An efficient symbolic out-of-core solution method for markov models. Technical Report CSR-03-08, School of Computer Science, University of Birmingham, Birmingham (2003)
15. Saad, Y., Van Der Vost, H.A.: Iterative solution of linear systems in the 20th century. J. Comput. Appl. Math. **123**, 1–33 (2000)
16. Banu, S., Vaideeswaran, D.: Performance Analysis on Parallel Sparse Matrix Vector Multiplication Micro-Benchmark Using Dynamic Instrumentation Pintool. Presented at the 2013, pp. 129–136 (2013)
17. Kwiatkowska, M., Mehmood, R.: Out-of-core solution of large linear systems of equations arising from Stochastic modelling. In: Hermanns, H., Segala, R. (eds.) Process Algebra and Probabilistic Methods: Performance Modeling and Verification. PAPM-PROBMIV, pp. 135–151. Springer, Berlin, Heidelberg (2002)
18. Mehmood, R.: Serial disk-based analysis of large stochastic models. In: Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.-P., Siegle, M. (eds.) Validation of Stochastic Systems: A Guide to Current Research, pp. 230–255. Springer, Berlin, Heidelberg (2004)
19. Mehmood, R., Crowcroft, J.: Parallel iterative solution method for large sparse linear equation systems. Technical Report Number UCAM-CL-TR-650, Computer Laboratory, University of Cambridge, Cambridge (2005)
20. Giles, M.B., Reguly, I.: Trends in high-performance computing for engineering calculations. Philos. Trans. R. Soc. A. **372**, 20130319 (2014)
21. Sodani, A., Gramunt, R., Corbal, J., Kim, H.S., Vinod, K., Chinthamani, S., Hutsell, S., Agarwal, R., Liu, Y.C.: Knights landing: second-generation Intel Xeon Phi Product. IEEE Micro. **36**, 34–46 (2016)
22. Eleliemy, A., Fayze, M., Mehmood, R., Katib, I., Aljohani, N.: Loadbalancing on parallel heterogeneous architectures: spin-image algorithm on CPU and MIC. In: EUROSIM 2016, The 9th Eurosim Congress on Modelling and Simulation. p. 6. Oulu (2016)
23. Alyahya, H., Mehmood, R., Katib, I.: Parallel sparse matrix vector multiplication on intel MIC: performance analysis. In: Smart Societies, Infrastructure, Technologies and Applications, SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, vol. 224, pp. 306–322. Springer, Cham (2018)
24. Björck, Å.: Numerical Methods in Matrix Computations. Springer International Publishing, Cham (2015)
25. Gander, W., Gander, M.J., Kwok, F.: Scientific Computing - An Introduction Using Maple and MATLAB. Springer Publishing Company, Incorporated (2014)
26. Koza, Z., Matyka, M., Mirosław, Ł., Poła, J.: Sparse matrix-vector product. In: Kindratenko, V. (ed.) Numerical Computations with GPUs, pp. 103–121. Springer International Publishing, Cham (2014)
27. Akhunov, R.R., Kuksenko, S.P., Salov, V.K., Gazizov, T.R.: Sparse matrix storage formats and acceleration of iterative solution of linear algebraic systems with dense matrices. J. Math. Sci. (United States). **191**, 10–18 (2013)
28. Cramer, T., Schmidl, D., Klemm, M., Mey, D.: OpenMP Programming on Intel Xeon Phi Coprocessors: An Early Performance Comparison, pp. 38–44. Marc@Rwth (2012)

29. Wang, E., Zhang, Q., Shen, B., Zhang, G., Lu, X., Wu, Q., Wang, Y.: High-Performance Computing on the Intel® Xeon Phi™. Springer International Publishing, Cham (2014)
30. Maeda, H., Takahashi, D.: Performance evaluation of sparse matrix-vector multiplication using GPU/MIC cluster. In: 2015 Third International Symposium on Computing and Networking. pp. 396–399 (2015)
31. Mehmood, R., Parker, D., Kwiatkowska, M.: An efficient BDD-based implementation of Gauss-Seidel for CTMC analysis. Technical Report CSR-03-13, School of Computer Science, University of Birmingham, Birmingham (2013)
32. Tang, Z., Huang, H., Jiang, H., Li, B.: MIC-based preconditioned conjugate gradient method for solving large sparse linear equations. In: Hung, J., Yen, N., Li, K.C. (eds.) Frontier Computing. Lecture Notes in Electrical Engineering, vol. 375. Springer, Singapore (2016)
33. Li, Z., Donde, V.D., Tournier, J.-C., Yang, F.: On limitations of traditional multi-core and potential of many-core processing architectures for sparse linear solvers used in large-scale power system applications. In: 2011 IEEE Power and Energy Society General Meeting, pp. 1–8. IEEE (2011)
34. Yan, D., Cao, H., Dong, X., Zhang, B., Zhang, X.: Optimizing algorithm of sparse linear systems on GPU. In: 2011 Sixth Annu. Chinagrid Conf. pp. 174–179 (2011)
35. Ye, F., Calvin, C., Petiton, S.G.: A study of SpMV implementation using MPI and OpenMP on Intel many-Core architecture. In: High Performance Computing for Computational Science—VECPAR 2014: 11th International Conference, Eugene, OR, USA, June 30–July 3, 2014, Revised Selected Papers, pp. 43–56. Springer International Publishing, Cham (2015)
36. Saule, E., Kaya, K., Catalyurek, U.V.: Performance evaluation of sparse matrix multiplication kernels on Intel Xeon Phi, ArXiv, Tech. Rep. arXiv:1302.1078, Feb (2013)
37. Maeda, H., Takahashi, D.: Parallel sparse matrix-vector multiplication using accelerators. In: Gervasi, O., Murgante, B., Misra, S., Rocha, A.M.A.C., Torre, C.M., Taniar, D., Apduhan, B.O., Stankova, E., Wang, S. (eds.) Computational Science and Its Applications—ICCSA 2016: 16th International Conference, Beijing, China, July 4–7, 2016, Proceedings, Part II, pp. 3–18. Springer International Publishing, Cham (2016)
38. Ahamed, A.-K.C., Magoules, F.: Iterative methods for sparse linear systems on graphics processing unit. In: 2012 IEEE 14th Int. Conf. High Perform. Comput. Commun. 2012 IEEE 9th Int. Conf. Embed. Softw. Syst. pp. 836–842 (2012)
39. Estebanez, A., Llanos, D.R., Gonzalez-Escribano, A.: Using the Xeon Phi platform to run speculatively-parallelized codes. Int. J. Parallel Prog. **45**, 225–241 (2017)
40. Dongarra, J., Gates, M., Haidar, A., Jia, Y., Kabir, K., Luszczek, P., Tomov, S.: HPC programming on Intel many-integrated-Core hardware with MAGMA port to Xeon phi. Sci. Program. **2015**, 1–11 (2015)
41. Chen, C., Yang, C., Tang, T., Wu, Q., Zhang, P.: OpenACC to intel offload: automatic translation and optimization. In: Communications in Computer and Information Science. pp. 111–120 (2013)
42. Davis, T.A., Hu, Y.: The University of Florida sparse matrix collection. ACM Trans. Math. Softw. **38**(1), 1–1:25 (2011)
43. Aziz Supercomputer, Top500. https://www.top500.org/site/50585

Chapter 17
Performance Characteristics for Sparse Matrix-Vector Multiplication on GPUs

Sarah AlAhmadi, Thaha Muhammed, Rashid Mehmood, and Aiiad Albeshri

17.1 Introduction

High-performance computing techniques can effectively enhance the performance of sparse linear equation systems, which have Sparse Matrix-Vector multiplication (SpMV) as the most important scientific computation unit [1]. Numerous important scientific, engineering and smart city applications require computations of sparse matrix-vector multiplication (SpMV) [2–6]. SpMV is a core computing part of many scientific and engineering applications such as finite element methods, signal processing, magneto-hydrodynamics, graphics processing, electrical power systems, data mining, graph analytics, and information retrieval [1, 7–11]. The widespread importance of sparse matrix computation has become research hotspots and brought about significant research endeavors into implementations based on modern-day parallel processors, mainly GPUs [1, 7, 10, 12, 13]. However, there are many challenges in computing SpMV such as the differences in sparsity patterns, that make such computations difficult.

The irregularities of sparse patterns result in a number of matrix representation issues [10]. Thus, there exists diverse sparse matrix storage layouts intended to exploit various sparsity designs and distinctive techniques for getting and manipulating matrix entries especially on GPUs. Direct or indirect improvements in data layout and data access pattern are solutions to obtain high throughput or indirect improvements. Furthermore, the SpMV performance is affected by the parallel

S. AlAhmadi (✉) · T. Muhammed · A. Albeshri
Department of Computer Science, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: salahmadi0084@stu.kau.edu.sa; m.thaha.h@ieee.org; aaalbeshri@kau.edu.sa

R. Mehmood
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: RMehmood@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_17

computing device platform being used [10]. SpMV algorithm that achieved good performance in one parallel device platform may not be as efficient as on other platforms due to the difference of the architecture and capabilities between the platforms.

This research will explore the SpMV and Jacobi iterative methods on GPUs with the aim to understand the performance bottlenecks and possibly address the limitations of the existing approaches. In Sect. 17.2, an overview of SpMV and Jacobi iterative techniques are provided. Section 17.3 explores the GPU architecture and performance characteristics of applications on GPU along with techniques to optimize the performance of SpMV. In Sect. 17.4, we explore the important storage formats for SpMV computations on GPU architecture. Finally, in Sect. 17.5 we analyze and discuss the performance of the notable storage formats using the identified performance characteristics and criterions.

17.2 SpMV and Iterative Methods

Sparse Matrix-Vector product (SpMV) is the most important process in scientific computing and engineering applications [1, 7, 8, 12, 14–17]. The performance of SpMV can be improved using parallel computing [1, 15, 16]. Sparse matrix is a matrix that have mostly zeros and very few non-zero elements [12, 17]. The processing of such matrices involves removal of the zeros elements to deal with just the non-zero (nnz) elements. The challenges involved in computing SpMV are numerous. Some of the major challenges are irregularity of the matrices, data transfer between host and device, load imbalance among the threads, memory access, and memory management (storage formats) [1, 7–9, 12, 14–17].

Iterative methods consist of a sequence of computations performed iteratively to produce approximate solutions that gradually reaches the accurate solution. They are, furthermore, partitioned into stationary methods (i.e., Jacobi) and non-stationary methods (i.e., conjugate gradient) like [17, 18]. In this work our attention will be on the stationary methods specifically the Jacobi iterative methods. Jacobi is an excellent candidate to be implemented on GPU although it is slower than other iterative methods, since it's inherently parallel.

Linear systems which have formula $Ax = b$ can be solved using Jacobi method as follows in each iteration compute $Ax = b$ as matrix-vector product, then test for convergence, and repeat until convergence. It involves partitioning of the matrix A into three parts: diagonal, upper-triangular, and lower-triangular portions [9, 19]. Thus, in matrix terms, the Jacobi method can be expressed as in Eq. (17.1):

$$x^k = D^{-1}(L + U)\, x^{k-1} + D^{-1} b \tag{17.1}$$

where k denotes the number of iterations, D is the diagonal entries, L is the lower-triangular matrix, and U is the upper-triangular matrix. Figure 17.1 depicts the Jacobi iterative technique and the SpMV involved. Many researchers have attempted

```
k=0
while not convergence do
  for i = 1 to n
    x_i = 0
    for j = 1 to n
      if j≠i
        x_i = x_i + a_i,j * x_j^(k)
      end
    end
  x_i^(k+1) = (b_i - x_i)/a_i,i
  end
  k = k + 1
end
```

Fig. 17.1 Algorithm for Jacobi iterative method depicting the SpMV operations involved

to improve the performance of iterative methods for sparse linear equation systems and SpMV computations [2, 17, 18, 20–27].

17.3 GPU: An Overview

In this section, we provide a brief overview of the general GPU architecture. We further discuss the GPU characteristics that affect the computational performance of the GPU and discuss various optimizations that enhance the performance.

17.3.1 Architecture

In the recent decade, GPUs are considered as a general-purpose processing unit instead of a mere graphics processing unit [7, 12, 28, 29]. GPU has attracted HPC researchers and has become popular in scientific computing due to its high computation capabilities, massive performance, effective usage of memory bandwidth, and the ability to accelerate existing large systems which have been implemented on other processors like CPU [7–9, 14, 30, 31].

Thus, GPUs become an important platform to implement sparse matrix computation to accelerate the performance of SpMV multiplication by processing them parallelly [8, 14, 15, 32]. Hence, many researchers have developed and optimized the existing algorithms to get best utilization out of these devices. Their speed can reach to Teraflops for single-precision calculations and half of this value for double-precision processing [9, 12]. Understanding GPU architecture is important for efficient utilization of the resources. However, the architectures are different for different GPU generations such as Kepler, Fermi, GeForce, PASCAL, and VOLTA. Different GPU families have been designed for different purposes such as

GeForce for graphics computation [33] and Tesla P100 for datacenters acceleration [34]. They differ largely on the parallel computing capabilities they have, thus the throughput performance delivered vary. Pascal, for example, is currently the most powerful architecture design for GPU. It turns a normal computer into a supercomputer and provides remarkable performance [33]. Tesla P100 belongs to the PASCAL family and it delivers a double-precision floating point about 5.3 TFLOPS, while Tesla V100 which belongs to the VOLTA architecture reaches to 7 TFLOPS [35, 36]. For all Nvidia versions among the last two decades along with its purposes, see [37]. A discussion on the Tesla P100 architecture can be seen in [34] and for Kepler architecture refer [33]. In addition to the architecture of the device, the selection of the best storage format for a given input matrix is a key issue [9, 12, 15].

Working with GPUs involves working with a heterogenous platform consisting hierarchies of computational units and memories. Figure 17.2 shows the different types of memories on such platforms and Fig. 17.3 shows the hierarchy of memory and computations on GPU. In general, GPU consists of an array of Streaming Multiprocessors (SM) that contains processing cores, and many types of memories such as registers and cache. The programming model for the GPU is single instruction multiple data (SIMD) applied into groups of 32 threads called warps. In subsequent sections, we further discuss about warps and its effects on the performance of SpMV.

Compute Unified Device Architecture (CUDA) is an API dedicated to GPU programming [38]. It depends on the C language and presents new possibilities for accelerating GPU kernels. Further, CUDA is developed to simplify and improve GPU programming and accelerating high-performance parallel computations [1, 7,

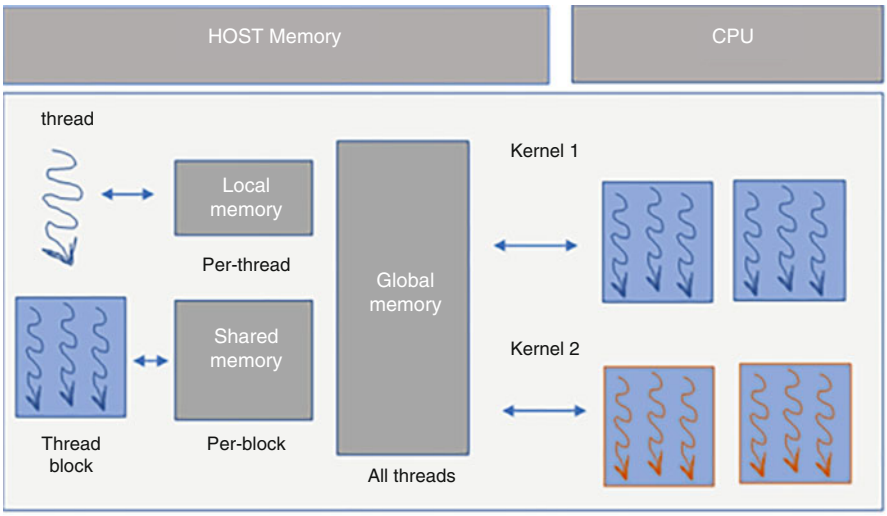


Fig. 17.2 Types of memory on heterogeneous platform

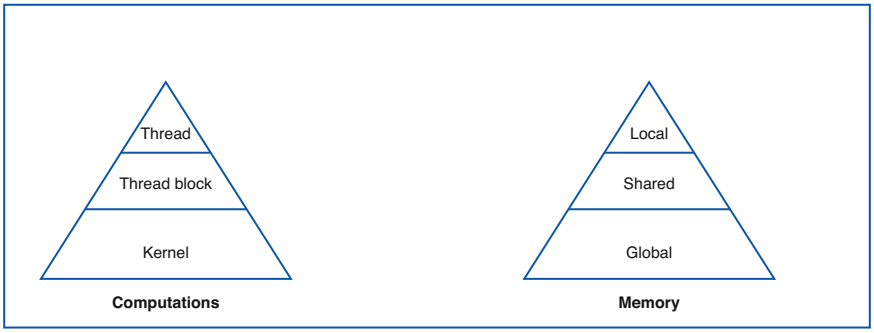


Fig. 17.3 Memory and computation hierarchies on GPU

38, 39]. CUDA views GPU as a grid of blocks where each block has a set of threads. The grid of blocks can be organized either as one-, two-, or three-dimensional computing units. A 2D grid example can be seen in [33].

17.3.2 Performance Characteristics: Discussion

The features that affect the performance of computations on GPUs can be broadly classified into three: (1) Execution configuration, (2) Memory throughput, and (3) Instruction throughput. In the following subsections, we discuss these performance characteristics in detail.

Execution Configuration These are the parameters that need to be configured at execution to improve the performance of GPU computations. The major execution configurations are:

1. *Active warps (Resource usage):* Registers and shared memory are critical components and need attention because it affects the number of active warps which in turn affects the GPU utilization. When the number of active warps is maximum, the GPU utilization is at maximum. The number of active warps can be maximized by good management of compute resources, registers, and shared memory. Reducing the number of registers utilized by a kernel results in higher warps being processed simultaneously. And when a thread block consumes more shared memory, fewer thread blocks are processed simultaneously by an SM. If the amount of shared memory used by each thread block is reduced, then more thread blocks can be processed simultaneously.
2. *Occupancy:* Occupancy is having enough warps to keep the device completely occupied. It is the ratio between active warps and maximum number of warps. We can compute occupancy by dividing the maximum number of threads per SM by 32. We can check it using CUDA occupancy calculator or the nvprof profiler. "*To enhance the occupancy, the thread block configuration needs to be*

resized or the resource usage needs to be readjusted to permit more active warps simultaneously and improve utilization of compute resources."

3. *Memory operations*: Load and store operations on the data should be measured to find the efficiency of the operation.

Memory Throughput Efficient utilization of the theoretical memory bandwidth of the GPUs improves the computational performance. The bandwidth utilization can be improved by considering the following factors:

1. *Data Transfer between Host and Device*: For the best performance of kernels, data transfers should be minimized between the host and device whenever possible and should be optimized by various techniques [40]. Moving more code from host to device is an efficient way to optimize the transfer process. In addition, every data transfer has an associated overhead, hence grouping many small transfers into single transfer reduces the overhead associated with each transfer and produces an overall better performance.
2. *Memory Access*: Memory access is an important factor that affects the overall performance of GPU applications. GPU has many types of memory as shown in Fig. 17.3. Scattered addresses in global memory need to be avoided; coalesced and aligned access can overcome throughput reduction. Thus, to increase global memory throughput, it is important to make the memory access transactions both aligned and coalesced. Coalesced access occurs within a warp scope when all the 32 threads in a warp use one memory transaction; more precisely, they access a contiguous chunk of memory [35]. Aligned access is to make sure the first address of a device memory transaction is a multiple of the transaction size, which usually has either 32, 64, or 128 bytes depending on the target device characteristics [35, 40]. Figure 17.4 shows the different memory access patterns.

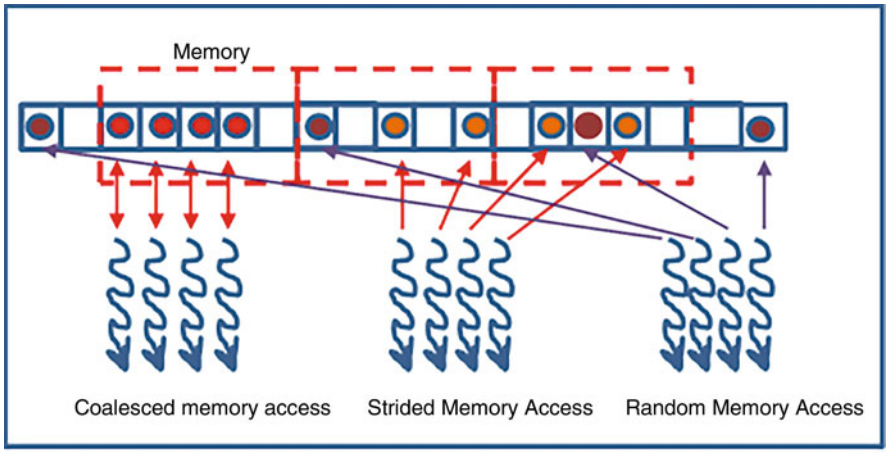


Fig. 17.4 Memory Access Patterns

The best access pattern is the coalesced access while the worse is the random access. As a rule, the more memory transactions required by a warp, the lower the memory throughput which leads to lower performance.

Instruction Throughput This describes the instruction optimizations that lead to the best performance. It can be summarized into three main perspectives as the following:

- *Arithmetic instructions*: Avoiding instructions that cost many operations per clock cycle such as mod operator. Avoid non-required conversions between datatypes.
- *Control Flow Instructions and Warp Divergence*: Control flow instructions (if, do, for, switch, while) can significantly impact the performance as it may cause warp divergence problem which degrades the instruction throughput [40]. Warp divergence occurs when the threads inside a warp have different execution paths. This conflicts with the fact that the GPU is a single instruction multiple data (SIMD) processor which would require all threads in a warp to execute one instruction. More precisely, threads on GPUs are organized as warps and each warp executes one instruction at a time for all threads inside that warp, each with its private data (SIMD). When control flow constructs are assigned to a warp, different branches might occur (e.g., some threads execute an if block when the condition is correct while others execute else block when the condition is wrong), which results in executing multiple instructions per warp. As a result, this process will be serially executed which results in idle threads in a warp as only one instruction will be executed at a time while next instruction will be loaded after the current instruction is finished [35, 41]. In other words, the instructions will be executed sequentially, thus the total number of instructions per warp increases. So, for best performance we should avoid different execution paths within a warp.
- *Synchronization:* It impacts the performance because of two reasons [35]. The first is the cost of the number of operations it requires (differs according to compute capability of the target device) while the second is forcing the multiprocessor to be idle while it is not required.

17.3.3 Performance Optimization Strategies

Performance optimization strategies according to [38] can be classified into three main categories, parallelism optimization, optimization of memory throughput, and optimization of instruction throughput. Each of these dimensions can be quantified using several metrics which can be measured using tools such as NVIDIA visual profiler, nvprof as command-line profiler tool, or by comparing the achieved throughput of a kernel to the corresponding peak theoretical throughput of the device to show how much improvement has been achieved by the kernel. The dimensions with their metrics have been explained in Table 17.1. For example, we can observe the memory operations in Nsight profiler using memory statistics menu.

Table 17.1 Performance dimensions of GPU Kernels

Performance dimension	Performance angles	Helping metrics
Memory optimization	• Aligned access	– gld_efficiency
	• Coalesced access	– gst_efficiency
	• Data transfer between host and device	– gld_transactions
		– gst_transactions
		– gld_throughput
		– gst_throughput
Instruction throughput	• Instruction throughput	– branch_efficiency
	• Warp divergence	– inst_per_warp
		– warp_execution_efficiency
Execution configuration optimization	• Occupancy	– achieved_occupancy
	• Tune grid blocks size	

Also, memory transaction metrics (**e.g., gld_transactions** and **gst_transactions**) can act as indirect indicators to measure the coalesced accesses. Higher memory transactions indicate a high probability of uncoalesced access. List of CUDA performance metrics collected by the nvprof can be found on [40].

17.3.4 Performance Optimization: Discussion

Looking for best kernel performance requires tuning multiple performance factors. It can be likened as a puzzle board that needs to compose many pieces to get a complete picture. We should look at different angles using multiple metrics to build better combination of performance aspects and get the best performance. However, there are possibilities of conflicts between these performance aspects that may degrade the overall performance even if we achieve high scores for individual factors. For example, getting more occupancy does not ensure the best performance, we can find that in some cases, low occupancy also has provided higher performance because there are other factors affecting the overall performance, for example, memory operations. In the same manner, getting high memory throughput does not equate to the best performance due to low efficiency these operations might have. As a matter of fact, memory efficiency is very important aspect to consider. It can be improved by changing the thread/block configuration. In general, there are some good tips for better configuration, keeping the block size always a multiple of the warp size (i.e., 32) and launching more blocks by setting the second dimension (block.y) to 1 to reduce the block size and obtain more blocks to launch. This will enhance the inter-block parallelism [35].

Therefore, enhancing the performance of GPU kernels can be done on multiple levels. One level might be focusing on exposing more parallelism by managing the

used resources such as registers and shared memory, or by controlling the occupancy (higher level of active warps at any given time) or any other aspects on this level as explained on previous section. Memory access management is another level to looking for. Optimization memory access to ensure coalesced and aligned access can immensely enhance the performance. The last level that we can work on is enhancement of instruction throughput by avoiding warp divergence or avoiding costly arithmetic operations such as mod operator. All these are examples and each level represents a separate field of optimizations. We can optimize on one level or more to find a good balance to get the better performance. Thus, looking for best GPU performance is a complicated process and requires checking the kernel from many angles and it does not depend on just a single metric.

17.4 SpMV Storage Formats and Computation Techniques

Data structures are a core aspect when dealing specially with SpMV and GPUs [8]. They have a strong impact on the performance of the algorithms that are used to solve SpMV [7–9, 15]. It represents the storage pattern of the input matrices in the memory and is responsible in providing the best data access [9, 14]. Numerous efforts have been made to improve storage formats specifically for SpMV on GPU and other architectures for iterative linear solvers on GPUs since sparse matrices show up in many applications which involve diverse computational patterns [16]. Accordingly, various storage formats have been proposed to facilitate the productivity and recovery of important information from the input matrix. The most prominent formats are the CSR, Coordinate format (COO), DIA, and ELL [39, 42]. In addition, some adaptation have been made to these basic formats such as CSR5 [43] and CSRNS [7], along with other hybrid Schemes [1, 8–10, 31, 42, 44]. CSR scheme is preferred always among comparable formats and it has been chosen because it is widely adopted, general-purpose storage format, and gives minimum memory accesses [7–9, 15, 16]. Furthermore, all prior storage formats are considered explicit while there are other storage schemes such as MTBDDs [45] which are considered implicit formats [17].

In this research, we have included our performance analysis on CSR, COO, ELL, DIA, HYB, and CSR5 schemes. The descriptions of these formats along with their analysis are explained in the next section.

17.5 Performance Analysis of Notable Sparse Storage and Computation Techniques

In this section, we explore several research efforts for SpMV optimization on GPU over the last years. We first describe how each scheme store the data and then we have defined the issues and limitations of each scheme. We have classified the

techniques according to the basic SpMV formats they are derived from. The basics storage formats are COO, CSR, ELL, DIA, and HYB.

17.5.1 Sparse Storage and SpMV Kernels: Qualitative Analysis

In the following subsection, we discuss some of the notable sparse formats and associated SpMV techniques [39, 44]:

- **The Coordinate (COO)** format is the most basic data structure to store a sparse matrix. It is made of three arrays: Row, Col, and Data to store the row indices, the columns indices, and the values of non-zero components, respectively.
- **The Compressed Sparse Row (CSR)** format is the most well-known format for sparse matrix storage. It comprises three arrays: RowPtr, Col, and Data to store row pointers to the offset of each row, indices of non-zero components, and values of non-zero components, respectively.
- **The ELLPACK (ELL)** structure stores a sparse matrix in two arrays: Data and Col. The array Data stores the values of non-zero components while Col array stores the columns indices of each non-zero component.
- **The Hybrid ELL/COO (HYB)** structure stores the greater part of non-zero components in ELL format and the rest of the non-zero components in COO. All non-zero components at the columns on the left of a threshold value are stocked in the ELL and the rest non-zero components are represented as COO format.
- **CSR5** proposed by [43] is an optimization of CSR format and it combines segmented sum technique for better load balance and compressed row data for better load/store operation efficiency. It is insensitive to sparsity structure of the input matrix. The matrix is partitioned into groups of 2D tails. These tails require extra information indicating their start index and columns indices, named as tail_ptr and tail_descriptor arrays, respectively. In addition, it has the CSR arrays val, col., and ptr. Thus, we have totally row_ptr, col_idx, val, tile_ptr, and tile_desc, where tile_desc further includes four arrays. tile_ptr works as row_ptr on CSR and it stores the row index of the first entry in each tile. Tail_desc has four different data structures, namely bit_flag, y_offset, seg_offset, and empty_offset arrays. These four arrays denote the start of each row inside the tiles, the address of the partial sum for each column, accelerating the segmented sum, and help the partial sums to find correct locations in y if the tile includes any empty rows. The illustrations of these schemes are shown in (Figs. 17.5, 17.6, 17.7, 17.8, 17.9 and 17.10).

Fig. 17.5 Original matrix

$$A = \begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 4 & 1 & 0 \\ 2 & 0 & 7 & 5 \\ 0 & 6 & 0 & 8 \end{bmatrix}$$

$$Var\_Arl = [1\ 3\ 4\ 1\ 2\ 7\ 5\ 6\ 8]$$
$$Col\_Arr = [0\ 2\ 1\ 2\ 0\ 2\ 3\ 1\ 3]$$
$$Ptr\_Arr = [0\ 2\ 4\ 7\ 9]$$

Fig. 17.6 CSR scheme

$$Val\_Arr = \begin{bmatrix} 1 & 3 & * \\ 4 & 1 & * \\ 2 & 7 & 5 \\ 6 & 8 & * \end{bmatrix} \qquad Col\_Arr = \begin{bmatrix} 0 & 2 & * \\ 1 & 2 & * \\ 0 & 2 & 3 \\ 1 & 3 & * \end{bmatrix}$$

Fig. 17.7 ELL scheme

$$Val\_Arr = \begin{bmatrix} * & 1 & * & 3 \\ * & 4 & 1 & * \\ 2 & 7 & 5 & * \\ 6 & 8 & * & * \end{bmatrix} \qquad offset\_Arr = [-2\ \ 0\ \ 1\ \ 2]$$

Fig. 17.8 DIA scheme

$$ELL \begin{cases} Val\_Arr = \begin{bmatrix} 1 & 3 \\ 4 & 1 \\ 2 & 7 \\ 6 & 8 \end{bmatrix} & Col\_Arr = \begin{bmatrix} 0 & 2 \\ 1 & 2 \\ 0 & 2 \\ 1 & 3 \end{bmatrix} \end{cases}$$

$$COO \begin{cases} Val\_Arr = [5] \\ Col\_Arr = [3] \\ Row\_Arr = [2] \end{cases}$$

Fig. 17.9 HYB scheme: ELL and COO

$$A = \begin{bmatrix} 3 & 0 & 2 & 1 & 0 & 5 \\ 0 & 4 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 5 & 3 & 0 & 0 & 7 \\ 0 & 1 & 0 & 8 & 0 & 2 \\ 9 & 5 & 7 & 6 & 7 & 0 \end{bmatrix}$$

$$val = \begin{bmatrix} 3 & 5 & 2 \\ 2 & 4 & 5 \\ 1 & 6 & 3 \end{bmatrix} \quad col\_index = \begin{bmatrix} 0 & 5 & 0 \\ 2 & 1 & 1 \\ 3 & 2 & 2 \end{bmatrix} \quad Tail\_0$$

$$val = \begin{bmatrix} 7 & 2 & 7 \\ 1 & 9 & 6 \\ 8 & 5 & 7 \end{bmatrix} \quad col\_index = \begin{bmatrix} 5 & 5 & 2 \\ 1 & 0 & 3 \\ 3 & 1 & 4 \end{bmatrix} \quad Tail\_1$$

$$m \times n = 6 \times 6 \quad nnz = 18 \quad \omega = 3 \quad \sigma = 3 \quad no\ of\ tails = (18/2) = 9$$

Fig. 17.10 CSR5 scheme

The main issues that should be taken into consideration regarding these basic formats are memory footprint in COO, coalesced access and thread mapping in CSR, and zero padding in ELL. Further, we shall illustrate the limitations of the selected techniques and analyze it with the performance evaluation criterion for SpMV and compare it with the performance characteristics of GPU. This comparison will show us the limitations of the existing techniques and how they are restricted to a few perspectives from a pool of GPU's performance considerations.

If we look at the performance criteria in each research, we can observe that the performance aspects covered on each technique is incomplete. Most of them focus on the speed of the technique while a few study the memory issues of their algorithms and seldom take into consideration the utilization rate of the GPU such as the occupancy rate of the device and the benefits from the massive parallelism provided by the target GPU. Table 17.2 illustrates the detailed performance data

for all the schemes discussed in this article and discusses the performance aspect considered in each research.

CSR and CSR Optimizations The main drawback of the scaler CSR (one thread per row) is the uncoalesced access of the data and indices arrays [1]. To rectify this issue, a vector CSR version is proposed (a warp per row) [1]. In addition, CSR is widely used for various types of sparse matrices, this flexibility introduces thread divergence problem especially for those sparse matrices with a variable number of non-zeros per row [1, 16]. This likely will cause many threads within a warp to be idle while waiting for the thread with the longest data to process. These drawbacks have been overcome by CSR vector version, but the performance of this version is strongly sensitive to the row size of the target matrix such that it is inefficient when rows have few non-zeros.

ELL and ELL Optimizations ELL accomplishes high performance on regular matrix structures (i.e., with an equivalent number of non-zeros on each row) [39, 46]. However, on irregular matrices unavoidably it leads to memory footprint inefficiency and misuse of computation (i.e., short rows make their thread inactive for most of the time) results in load imbalance. The granularity of ELL SpMV on GPU is one thread per row. Nevertheless, it implicates potential space wastage with the way that all rows are zero-padded to length $N_{\max}$. Subsequently, this configuration is most productive when the variance of non-zeros among rows is small [10].

AdELL+ SpMV kernel proposed by [46] is an improvement of ELL format and it is also kind of hybrid format that combines ELL and CSR. It outperforms the comparable kernels in terms of speed of execution measured on GFLOPS for both regular and irregular matrices. They have discussed memory bandwidth but without comparison with others, so we cannot decide about amount of improvements done on this point. They also have measured memory footprints compared with CSR structure and it has achieved less footprints than CSR.

HYB Single storage configuration only provides the best performance only in limited situations which gave birth to the idea of hybrid formats. HYB format, for example, is the first hybrid format consisting of COO and ELL formats to overcome the sensitivity to the sparsity structures in both ELL and COO. It successfully achieved good performance and is considered as one of the best formats especially on the unstructured matrices. However, it has higher costs including high level of data organization, has complicated program logic, and costs time in terms of memory transfer [12].

SHEC [10] is another segmented hybrid format that consists of ELL and CSR (vector version). They combine the advantage of ELL granularity (i.e., one thread per row) and CSR granularity (i.e., one warp per row). SHEC is intended for further improvements on the throughput of SpMV and specially to lessen the memory footprint on GPUs.

Another hybrid scheme has been proposed in [13] which combines DIA with the ELLPACK structure. This combination isolates the diagonal elements of the

sparse matrix using the DIA scheme while the residual elements are stored in ELLPACK format. This is immensely beneficial for iterative methods, specifically Jacobi iteration because it uses the diagonal values in its calculation, so the isolation on the proposed format will give faster access to the diagonal elements. However, the performance is limited compared to ELLPACK and it is highly efficient for those matrices having a relatively dense diagonal band [13].

CSR5 [43] have been introduced as a storage format that is based on segmented CSR. The authors considered computation intensity factor to measure their performance compared with others. It significantly improves the load imbalance problem that CSR suffer from. CSR5 is a complex storage format and requires several arrays. These arrays involve more memory access operations (many load operations) and large memory space to load this information which may affects the total memory efficiency. Poor resource management lead to less GPU utilization since it effects number of blocks and warps working concurrently which subsequently affects rate of the device occupancy. Furthermore, the memory bandwidth measurements are not provided and the technique is not space efficient due to the large number of arrays used. In addition to its complexity, CSR5 has significant overheads due to the preprocessing process such as the matrix transpose operations and transformation from CSR **val** and **col** arrays into CSR5 arrays

17.5.2 Performance Comparison

Table 17.2 compares the six considered SpMV kernels which are CSR (scalar vector), COO, DIA, ELL, HYB, and CSR5. The comparison is in terms of the used GPU device, peak theoretical values of performance and memory bandwidth, and the achieved performance and memory throughput. In addition, we provide the name of the matrix benchmark suites used in the experiments. Some have used wide variety of real application matrices derived from finite element method-based modeling, linear programming, circuit simulation, and connectivity graphs from partial web crawls. It should be noted that the notable CSR version is CSR scalar which has granularity of one thread per row. The comparison includes several experiments from different studies of the selected structures. In this article, we have considered experiments of double-precision computations and unstructured matrices excluding single-precision and structured matrices except for DIA and ELL schemes. We have considered the structured matrices for DIA and ELL because they are dedicated for such matrices. GFLOPS refers to performance throughput. The calculation for single-precision and double-precision flops are different. The formula to calculate the peak value of double precisions is given in Eq. (17.2).

$$2 \times \left[\text{multiply add}\right] \times \left[\#\text{of multiprocessors}/8\right] \times \left[\text{processor clock}/1000\right] \quad (17.2)$$

Table 17.2 Comparison of SpMV Kernels

	GFLOPS			Memory bandwidth	
Tech. name	Device	Peak GFLOPS	Obtained GFLOPS (MAX)	Peak memory bandwidth	Effective bandwidth (MAX)
(COO 1990) [37]	GTX 280 [35]	77.76	4	141.7	58
	GTX 285 [38]	88.56	5	159.0	–
(CSR Scalar 1990) [37]	GTX 280 [35]	77.76	4	141.7	55
	GTX 285 [38]	88.56	4.2	159.0	–
	GTX 980 [39]	144.1	18	141.7	–
(ELL 1985) [37]	GTX 280 [35]	77.76	13.5	141.7	140
	GTX 285 [38]	88.56	15	159.0	–
(DIA 1989) [37]	GTX 280 [35]	77.76	16.7	141.7	141
	GTX 285 [38]	88.56	18.2	159.0	–
(HYB 2008) [35]	GTX 280 [35]	77.76	14	141.7	141
	GTX 285 [38]	88.56	15.7	159.0	–
	GTX 980 [39]	144.1	15		–
(CSR5 2015) [39]	GTX 980 [39]	144.1	27	224	–

The peak GFLOPS discussed in this article is either calculated using Eq. (17.2) or from the device specifications given on the website, or is mostly reported in various researches. The obtained performance throughput measures the number of floating point operations per second, and it is calculated by dividing the required arithmetic operations by the average execution time [42]. Peak memory bandwidth is clearly defined on the device specifications, otherwise it can be calculated using Eq. (17.3).

$$(\text{Memory clock} \times \text{Bus Width}/8) \times \text{GDDR type multiplier} \tag{17.3}$$

GDDR multiplier values vary according to memory type. For GDDR3, GDDR5, and GDDR5X, it is 2, 4, and 8, respectively. Division by 8 is to change from bit to byte. Effective bandwidth is defined as the total number of bytes written/read by all threads divided by the average execution time [42].

In [39], they have implemented the basics formats for structured and unstructured matrices with single and double-precision computations. In addition, they have considered experiments with and without cache. They have considered the GPU performance measured in GFLOPS as well as memory bandwidth measured in GB/s as performance aspects. They have evaluated the performance results using single and double-precision floating points and measured the performance enhancement. However, they do not consider peak performance and peak memory bandwidth to measure the achievable performance compared with device capabilities. If we compare the achieved results with the peak values, we observe they have lower performance compared to the device capabilities as stated in Table 17.2.

In [39, 42], they have the same experiments on different devices with slight enhancements as compared to [42]. SpMV is a memory-bounded computation and

hence they did not achieve the peak performance of the used devices [10, 39, 46–48]. More precisely, if we study the performance characteristics that have been discussed in Sect. 17.3 for the selected kernels we can observe many limitations. Coalesced memory access, for example, is a difficult issue on sparse matrix computations because different storage schemes require many pointers that point to the address of the first element of the blocks, slices, and individual rows. However, the need for these addresses mean the need for more arrays (at least one beside the data array) which would result in loading more arrays into the device global memory. This would increase the memory transactions which may degrade the performance if the accessing pattern is not coalesced. Furthermore, instead of a single array, all the arrays should have a coalesced access to ensure better performance. COO, DIA, ELL, and HYB formats are fully coalesced [1]. On the other hand, CSR does not provide a coalesced access either for the data array nor to the other arrays. CSR5 supports memory coalesced access by accessing the data and column arrays in column-major order instead of row-major order as seen in the classic CSR.

Warp divergence is another performance characteristic and likely to occur on CSR. It results in load imbalance between threads; however, it is significantly less in CSR5 by dividing the elements into fixed-size tails. Moreover, other performance aspects such as instruction throughput, occupancy, block-thread heuristics, number of resources used, and other performance aspects are not considered. For the best performance and device utilization we should include a combination of performance characteristics to evaluate the performance which most researches lack of. Furthermore, the properties of GPU architecture included in the experiments have significant impact on the achieved performance as we have seen in our comparison. However, even with different GPU devices, the achieved SPMV performance is low as compared to the high throughput each device can provide.

17.6 Conclusion

In this chapter, we discussed the performance of SpMV on GPU architectures. We provided an architectural overview of GPU devices and defined the performance dimensions of GPU computations. We explored the performance of a few major existing sparse matrix storage formats. We concluded that there is lack of performance aspects considered during the evaluation of the existing SpMV algorithms, specifically to measure the memory throughput achieved by the SpMV computations. Since SpMV computations are memory-bound, the achieved performance should be compared to the peak theoretical bandwidth of the GPUs. We conclude that to achieve better performance analysis a combination of performance aspects/criterion should be noted. In addition, the performance of SpMV on different GPU device architecture varies. Hence, a comprehensive and standard set of performance characteristics need to be used by the researchers while comparing and analyzing SpMV on GPUs.

References

1. Yang, W., Li, K., Li, K.: A hybrid computing method of SpMV on CPU–GPU heterogeneous computing systems. J. Parallel Distrib. Comput. **104**, 49–60 (2017)
2. Mehmood, R., Lu, J.A.: Computational Markovian analysis of large systems. J. Manuf. Technol. Manag. **22**, 804–817 (2011)
3. Mehmood, R., Meriton, R., Graham, G., Hennelly, P., Kumar, M.: Exploring the influence of big data on city transport operations: a Markovian approach. Int. J. Oper. Prod. Manag. **37**, 75–104 (2017)
4. Mehmood, R., Alturki, R., Zeadally, S.: Multimedia applications over metropolitan area networks (MANs). J. Netw. Comput. Appl. **34**, 1518–1529 (2011)
5. Mehmood, R., Graham, G.: Big data logistics: a health-care transport capacity sharing model. Proc. Comput. Sci. **64**, 1107–1114 (2015)
6. Altowaijri, S., Mehmood, R., Williams, J.: A quantitative model of grid systems performance in healthcare organisations. ISMS 2010—UKSim/AMSS 1st International Conference on Intelligent Systems. Model. Simul. 431–436 (2010)
7. Huan, G., Qian, Z.: A new method of sparse matrix-vector multiplication on GPU. In: International Conference on Computer Science and Network Technology, pp. 954–958 (2012)
8. Hassani, R., Fazely, A., Choudhury, R.-U.-A., Luksch, P.: Analysis of sparse matrix-vector multiplication using iterative method in CUDA. In: 2013 IEEE Eighth International Conference on Networking, Architecture and Storage, pp. 262–266 (2013)
9. Cheik Ahamed, A.-K., Magoulès, F.: Efficient implementation of Jacobi iterative method for large sparse linear systems on graphic processing units. J. Supercomput. **73**, 3411–3432 (2017)
10. Adhianto, L., Banerjee, S., Fagan, M., Krentel, M., Marin, G., Mellor-Crummey, J., Tallent, N.R.: HPCTOOLKIT: tools for performance analysis of optimized parallel programs. Concurr. Comput. Pract. Exp. **22**, 685–701 (2010). http://hpctoolkit.org
11. Brahme, D., Mishra, B.R., Barve, A.: Parallel sparse matrix vector multiplication using greedy extraction of boxes. In: 2010 International Conference on High Performance Computing, pp. 1–10 (2010)
12. Ahamed, A.-K.C., Magoules, F.: Fast sparse matrix-vector multiplication on graphics processing unit for finite element analysis. In: 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems, pp. 1307–1314 (2012)
13. Guo, P., Wang, L., Chen, P.: A performance modeling and optimization analysis tool for sparse matrix-vector multiplication on GPUs. IEEE Trans. Parallel Distrib. Syst. **25**, 1112–1123 (2014)
14. Guo, P., Wang, L.: Auto-tuning CUDA parameters for sparse matrix-vector multiplication on GPUs. In: Proceedings—2010 International Conference on Computational and Information Sciences, ICCIS 2010, pp. 1154–1157 (2010)
15. Merrill, D., Garland, M.: Merge-based parallel sparse matrix-vector multiplication. In: International Conference for High Performance Computing, Networking, Storage and Analysis, SC 16, pp. 678–689 (2016)
16. Hou, K., Feng, W.C., Che, S.: Auto-tuning strategies for parallelizing sparse matrix-vector (SpMV) multiplication on multi- and many-core processors. In: Proceedings—2017 IEEE 31st International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2017, pp. 713–722 (2017)
17. Mehmood, R., Crowcroft, J.: Parallel Iterative Solution Method for Large Sparse Linear Equation Systems. UCAM-CL-TR-650. University of Cambridge, Computer Laboratory (2005)
18. Mehmood, R.: Disk-Based Techniques for Efficient Solution of Large Markov Chains. Computer Science, University of Birmingham (2004)
19. Barrett, R., Berry, M., Chan, T.F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., van der Vorst, H.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. Society for Industrial and Applied Mathematics, Philadelphia (1994)

20. Eleliemy, A., Fayez, M., Mehmood, R., Katib, I., Aljohani, N.: Loadbalancing on parallel heterogeneous architectures: spin-image algorithm on CPU and MIC. In: 9th EUROSIM Congress on Modelling and Simulation. EUROSIM (2016)
21. Kwiatkowska, M., Mehmood, R.: Out-of-Core solution of large linear Systems of Equations Arising from stochastic modelling. In: Process Algebra and Probabilistic Methods: Performance Modeling and Verification, pp. 135–151. Springer, Berlin (2002)
22. Kwiatkowska, M., Mehmood, R., Norman, G., Parker, D.: A symbolic out-of-core solution method for Markov models. Electr. Notes Theor. Comput. Sci. **68**, 589–604 (2002)
23. Mehmood, R.: A Survey of Out-of-Core Analysis Techniques in Stochastic Modelling. University of Birmingham, UK (2003)
24. Mehmood, R.: Serial disk-based analysis of large stochastic models. In: Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.-P., Siegle, M. (eds.) Validation of Stochastic Systems: A Guide to Current Research, pp. 230–255. Springer, Berlin (2004)
25. Mehmood, R., Crowcroft, J., Elmirghani, J.M.H.: A parallel implicit method for the steady-state solution of CTMCs. In: 14th IEEE International Symposium on Modeling, Analysis, and Simulation, pp. 293–302 (2006)
26. Mehmood, R., Parker, D., Kwiatkowska, M.: An Efficient BDD-Based Implementation of Gauss-Seidel for CTMC Analysis. University of Birmingham, UK (2003)
27. Mehmood, R., Parker, D., Kwiatkowska, M.: An Efficient Symbolic Out-of-Core Solution Method for Markov Models., University of Birmingham, UK (2003)
28. Magoulès, F., Ahamed, A.-K.C.: Alinea: an advanced linear algebra library for massively parallel computations on graphics processing units. Int. J. High Perform. Comput. Appl. **29**, 284–310 (2015)
29. Muhammed, T., Mehmood, R., Albeshri, A., Katib, I.: UbeHealth: a personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities. IEEE Access. **6**, 32258–32285 (2018)
30. Owens, J.D., Houston, M., Luebke, D., Green, S., Stone, J.E., Phillips, J.C.: GPU computing. Proc. IEEE. 879–899 (2008)
31. Fevgas, A., Daloukas, K., Tsompanopoulou, P., Bozanis, P.: Efficient solution of large sparse linear systems in modern hardware. In: 2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA), pp. 1–6 (2015)
32. Kirk, D.B., Hwu, W.M.W.: Programming Massively Parallel Processors: A Hands-on Approach (2013)
33. Cheng, J., Grossman, M., McKercher, T.: Professional CUDA C Programming. Wiley, New York (2014)
34. NVIDIA: Pascal GPU Architecture | NVIDIA. https://www.nvidia.com/en-us/data-center/pascal-gpu-architecture
35. NVIDIA: NVIDIA Tesla P100 Whitepaper (2016)
36. NVIDIA: NVIDIA Tesla V100. https://www.nvidia.com/en-us/data-center/tesla-v100/?ncid=van-tesla-v100
37. NVIDIA: History of NVIDIA – From Graphics Cards to Mobile Processors. http://www.nvidia.co.uk/object/corporate-timeline-uk.html
38. Nvidia: Nvidia CUDA C Programming Guide Version 4.2 (2012)
39. Bell, N., Garland, M.: Efficient Sparse Matrix-Vector Multiplication on CUDA (2008)
40. Nvidia: Profiler User's Guide
41. Saad, Y.: SPARSKIT: A Basic Tool Kit for Sparse Matrix Computations Version 2 (1994)
42. Bell, N., Garland, M.: Implementing sparse matrix-vector multiplication on throughput-oriented processors. In: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis—SC '09, p. 1 (2009)
43. Liu, W., Vinter, B.: CSR5: An Efficient Storage Format for Cross-Platform Sparse Matrix-Vector Multiplication. Arxiv, Ithaca, NY (2015)
44. Guo, P., Lee, C.W.: A performance prediction and analysis integrated framework for SpMV on GPUs. In: Procedia Computer Science, pp. 178–189. The Author(s), (2016)

45. Fujita, M., McGeer, P.C., Yang, J.C.-Y.: Multi-terminal binary decision diagrams: an efficient data structure for matrix representation. Formal Meth. Syst. Design. **10**, 149–169 (1997)
46. Maggioni, M., Berger-Wolf, T.: Optimization techniques for sparse matrix-vector multiplication on GPUs. J. Parallel Distrib. Comput. **93-94**, 66–86 (2016)
47. Filippone, S., Cardellini, V., Barbieri, D., Fanfarillo, A.: Sparse matrix-vector multiplication on GPGPUs. ACM Trans. Math. Softw. **43**, 1–49 (2017)
48. Williams, S., Oliker, L., Vuduc, R., Shalf, J., Yelick, K., Demmel, J.: Optimization of sparse matrix-vector multiplication on emerging multicore platforms—long version. Parallel Comput. **35**, 178–194 (2009)

Chapter 18
HPC-Smart Infrastructures: A Review and Outlook on Performance Analysis Methods and Tools

Thaha Muhammed, Rashid Mehmood, Aiiad Albeshri, and Fawaz Alsolami

18.1 Introduction

High-performance computing (HPC) plays a vital role in driving transformations across various smart-city infrastructures such as healthcare, agriculture, environment, and other infrastructures [94]. It is a vital cog in autonomous adaption of urban infrastructure to various events and stimuli (e.g., severe hurricane, high traffic due to accidents). HPC is a major component in developing phenomenal computationally intensive models for various smart-city infrastructures.

Driving high efficiency from shared-memory and distributed-memory HPC systems have always been challenging. Big data and HPC convergence, system heterogeneity, cloud computing, and many other developments have increased the complexities of HPC systems [36, 51, 77, 108, 124]. There are increasing pressures on energy efficiency for developing exascale computers and therefore development of highly efficient HPC applications and systems has become essential.

Various performance analysis tools exist that help in improving the performance and efficiency of HPC scientific applications and increase their potential. Performance analysis is a crucial part in the development of the HPC applications. Performance optimization is not just identifying the bottlenecks in the code but also identifying the causes of bottlenecks and the required changes that need to be made to the parallel applications [99]. This requires more advanced tools such as hardware performance counters. Diagnosing the problems manually requires

T. Muhammed (✉) · A. Albeshri · F. Alsolami
Department of Computer Science, FCIT, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: m.thaha.h@ieee.org; aaalbeshri@kau.edu.sa; falsolami1@kau.edu.sa

R. Mehmood
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: rmehmood@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_18

deep knowledge about the architecture, hardware of the system, and the compiler. Performance analysis is important to determine the different optimization strategies for the same application on different HPC platforms such as GPU, MIC, cloud, and MPI-based grids.

Corresponding to debugging and testing, performance analysis and optimization of HPC applications are vital stages in the development cycle. It is a crucial condition for assuring efficient use of costly and limited resources. The performance analysis phase evaluates the actual performance (speed of computation, throughput, and resource consumption) on a given platform with regard to memory, storage, network, and runtime. Moreover, it has to identify improvements and reduction in the usage of resources.

This paper reviews the performance analysis tools and techniques for HPC applications and systems. The contributions of this article can be summarized below.

1. A review of the tools for the performance analysis of HPC applications. The works on the HPC performance analysis are numerous and we do not claim to be exhaustive in this paper.
2. A discussion on the performance of various HPC applications on a number of HPC platforms.
3. A discussion on the common HPC applications used by the researchers and HPC benchmarking suites for analysis.
4. A qualitative comparison of various tools used for the performance analysis of HPC applications is provided.
5. A discussion on the future research directions and issues.

The rest of the paper is organized as follows. Section 18.2 describes various Benchmark toolkits and various HPC applications that are used for the performance analysis. Section 18.3 presents existing work by various researchers in analyzing the performance of various HPC applications. Section 18.4 provides a qualitative comparison of various tools used for HPC application performance analysis. Future research issues and directions are provided in Sect. 18.5. Finally, Sect. 18.6 concludes the paper.

18.2 HPC Applications and Benchmarking Suites

In this section, we discuss various HPC-based applications from various domains which are prone to performance problems. Table 18.1 summarizes various HPC applications that are used in various application domains. We shall discuss some major domains in which HPC is a necessity and is used abundantly.

– **Automobile and Aeronautics:** This field has a lot of simulation and modeling, model prediction and verification including probabilistic modeling, computer aided drawing, graphic designing, design automation, the design of structures,

Table 18.1 A summary and comparison of commonly used HPC applications

Application	Domain	Language	Developers	OS	Open
BigDFT	Chemistry	F90	Genovese et al. [42]	Li/Un	Yes
Bifrost	Atmosphere	F90	Gudiksen et al. [46]	Li/Un	No
ChaNGa	Cosmology	Charm++	Jetley et al. [55]	Li/Un	Yes
COSMO	Weather	C++	CINECA	Li/Un	Yes
CORSIKA	Astrophysics	F77/F90	Heck et al. [50]	Li/Un	No
ECHAM/MESSy	Environment	F77/F90	Jöckel et al. [57]	Li/Un	No
EUTERPE	Fusion	C++/C	Saez et al. [110]	Li/Un	No
Gamess	Material Science	F77	Schmidt et al. [45, 112]	Li/Un	No
IBM WATSON	Graph Analysis	C++	IBM	Li/Un	No
IMPACT-T	Math. modeling	F90	Qinag et al. [107]	Li/Un	Yes
Jacobi2D	Math. modeling	Charm++	–	Li/Un	Yes
LIBMESH	Math. modeling	C/C++	Kirk et al. [64]	Li/Un	Yes
MAESTRO	Astrophysics	F90	Nonaka et al.[100]	Li/Un	Yes
MILC	Quantum Theory	C/C++	Bailey et al. [20]	Li/Un	Yes
MP2C	Particle collision	F90	Freche et al. [39]	Li/Un	No
NAMD	Chemistry	Charm++	Bhatele et al. [23]	Li/Un	Yes
NQueens	Backtracking	C/C++	–	Li/Un	Yes
OpenFOAM	Fluid dynamics	C++	Jacobsen et al. [54]	Li/Un	Yes
Paratec	Quantum theory	C/C++	Pfrommer et al. [104]	Li/Un	No
PEPC	Gravitation	F2003	Gibbon [44]	Li/Un	Yes
ProFASI	Protein structure	C++	Irbäck et al. [52]	Li/Un	No
PRISM	Probab. modeling	Java/C	Kwiatkowska et al. [72]	Li/Win	Yes
Quantum expresso	Molecular structure	F90	Giannozzi et al. [43]	Li/Win	Yes
SIMONA	Nano science	C++	Strunk et al. [116]	Li/Un	No
SMMP	Protein structure	F90	Meinke et al. [90]	Li/Un	Yes
SPECFEM3D	Wave propagation	F90/C	Dimitri et al. [67]	Li/Un	Yes
Sweep3D	Material science	F77	Wylie et al.. [128]	Li/Un	Yes
YALES2	Combustion	C++/F90	Moureau et al. [91]	Li/Un	No
WIEN2K	Chemistry	F90	Schwarz et al. [113]	Li/Un	No

Li Linux, *Un* Unix, *Win* Windows

automated plan building, analysis of design, and concrete modeling [66, 73, 98, 130].

- **Astrophysics and quantum physics:** A lot of applications based on physics, especially on quantum physics and astrophysics, has very large computations as they receive a large input data. Load balancing of spin-image algorithm on CPU and MIC has been studied in [22, 34].
- **Biosciences:** Biosciences including bioinformatics have a large number of programs that require computation including the mathematical modeling of diseases. It also has issues with memory management. See, for example, [7].
- **Earth sciences:** A large number of earth related activities such as earthquake prediction, monitoring, weather prediction, and prediction of climate change due

to global warming needs high computation [61, 109]. These applications are highly data intensive and take days to run on serial machines.

- **Electronics:** The design and analysis of electronic components have a high computation due to the simulation and modeling before the actual production [117, 129]. Other things include the static timing analysis and lithography.
- **Material sciences:** Material science includes modeling of nanoscale particle, the modeling behavior of nanoscale particle, and modeling of molecules and chemical reactions [2, 74]. These require a lot of computation and memory.
- **Computational fluid dynamics (CFD):** CFD is used to model the flow of fluid around and within an object by solving mathematical equations governing the flow with the help of numerical methods. It is an inter-disciplinary domain and has applications in multiple domains. Complex flow phenomenon can be simulated with CFD. However, it requires massively parallel supercomputers to run the simulations efficiently and effectively [41, 105].
- **Graph computations:** Graphs are extensively used combinatorial tools in computing. They are used for representing sparse matrices, assist load balancing in computations [16], model molecular structures, traffic networks [16], and social media networks [8, 119], and distribution networks. It is also used in bioinformatics, business-analytics, and city planning. As graphs grow larger in size, we require powerful computational techniques for effective processing.
- **Computational and artificial intelligence (AI):** AI has become a fundamental technique for developing smarter algorithms and solutions in all scientific computations domains. Training deep learning and machine learning-based models require large computing power. For example, AI in healthcare networked systems [95] and educational systems [88] provides a better quality of service (QoS) and experience to the users. Deep learning has also been used to forecast traffic conditions for smart cities [14], and there are numerous other applications of AI, machine and deep learning. Probabilistic methods have also been used for computational intelligence, see, e.g., [71, 80] and the references therein. Solution of sparse linear equation system is an important part of such computational intelligence techniques. Solving sparse linear equation system mainly consists of sparse matrix vector multiplication which requires efficient utilization of parallel devices and this is discussed next.
- **Linear algebra and matrix computations:** A large number of scientific domains require linear algebra and matrix computations, such as dense or sparse matrix matrix multiplications (MMM), dense matrix vector products (MVPs), and sparse matrix vector products (SpMV) [3]. Application of HPC for dense linear algebra (MVP/MMM) can be seen in [17, 122, 123]. Work on efficiently utilizing parallel devices for SpMV can be seen in [11, 12, 69, 70, 79–82, 85, 86].
- **Big data:** Big data refers to "the emerging technologies that are designed to extract value from data having four vs characteristics; volume, variety, velocity and veracity" [87]. Big data technologies are being used in many application areas that require HPC to address big data challenges, see, e.g., [5, 83, 88, 95, 118, 119]. There are many ongoing efforts on the convergence of HPC and big data [36, 108, 124].

Table 18.2 A summary and comparison of benchmarking suites for HPC applications

Name	Developers	Benchmark type	Supports	Language
DEISA	EUS	Real apps	Cloud+MPI	C
HPC challenge benchmark	DARPA	Micro	MPI+OpenMP	C
Iometer	Intel	I/O	All network environments	C++
LINPACK	Dongarra et. al.	Kernel	MPI+OpenMP	Fortran
NAS parallel benchmark (NPB)	NASA	Kernel	MPI	C/C++
NPB multi-zone(NPB-MZ)	NASA	Kernel	MPI+OpenMP	C/C++
PARSEC	Princeton Univ.	Kernel	Multi-threaded SMA	C/C++
PerfKitBenchmarker	Google	Kernel	Cloud environment	Python
PMaC HPC	SciDac PERC	Micro	MPI+OpenMP	C++
Rodinia	Kevin Skadron	Real apps	CUDA/OpenMP	C/C++
STREAM	UOV	Kernel	MPI	C++/F90
VMmark	VMware	Virtual machine	Virtual machines & cloud	C/C++

– **Smart Cities, societies, and infrastructure:** Smart cities are driven by the rapid advancements in ICT technologies. These ICT developments have given rise to the integration and convergence of digital and physical systems such as computing, communications, big data, transport, healthcare, and city operations [6, 8, 10, 15, 16, 63, 83, 84, 89, 94, 95, 111, 118, 119, 124]. See, e.g., [88] for background on smart cities and societies.

Researchers have used applications from the discussed domains to study the performance of these applications in high-performance environments. Several works analyze the performance of applications from discussed domains using various known benchmarks. Table 18.1 lists some of the important applications that use HPC and Table 18.2 provides some of the major benchmarks used. Performance analysis tools have been used to analyze the applications to detect the bottlenecks in the code. We shall discuss these tools in later sections along with the review of earlier research.

18.3 Performance Analysis of HPC Applications: Literature Review

18.3.1 Performance Analysis Metrics (Theoretical)

Carrington et al. [27] analyze the metrics used for evaluating the performance of HPC applications. They mainly evaluate a simple synthetic metric, a linear

combination of various single metric with weights. They also test a metrics derived by convolving an application transfer function with the system performance data obtained using any one single simple metric which is also known as predictive metrics. The authors evaluate the performance of ten Department of Defense high-performance computing modernization applications (HPCMP) [31]. Of the ten selected application, five of the applications are workload dependent and the other five are workload independent. Each of these ten applications was run on ten target systems with a distinct architecture. The simple benchmarks such as LINPACK [31], STREAM [78], and HPC challenge [76] have a weak correlation to performance and hence the authors additionally use synthetic benchmarks in combination with prediction and performance modeling framework [48]. A transfer function is applied to the test result by the prediction model that enables the representation of multiple categories using one single metric. In the simple metric scheme, the metric from a single benchmark is used, whereas in predictive benchmark they use a set of single benchmark metrics along with a real-time trace of the application. Simple metrics is modeled as follows:

$$T'(A, B) = \frac{L(A)}{L(A_o)} \cdot T(A_o, B) \tag{18.1}$$

where $T'(A, B)$ is the predicted clock time for application B on system A, $L(A)$ is the result for a specific single benchmark for system L, A_o denotes the base system benchmark, and $T(A, B)$ is the measured wall clock time for A on B. The errors reported are calculated as

$$\%\, Error = \frac{T'(A, B) - T(A_o, B)}{T(A, B)} \tag{18.2}$$

For predictive metrics, the authors use a tracer such as MetaSim tracer [28] for dynamic tracing of the base station and synthetic probes are used for measuring the rates for each operation on a target system. The MetaSim convolver [114] divides the execution operation count by operation rate to achieve execution type for current basic block per operation. After experimentation, the authors conclude that the correlation between the metric and real-time performance is higher than simple metrics.

18.3.2 HPC on the Clouds

Gupta et al. [47] provide an evaluation and comparison between the performance of HPC applications on the cloud and on traditional HPC systems such as super-computers and clusters. They also answer questions such as which HPC application is suitable for cloud, when is it suitable to choose cloud to run HPC application, and what application can be run on the cloud. The authors grade the performance

of a number of selected applications on a number of the platform including supercomputer, a different type of clouds and clusters. The authors recognize various bottlenecks and the correlation between the characteristics and performance of the HPC application. The authors use three different benchmarks to analyze the performance of the HPC application. These are NAS parallel benchmark [97], a benchmark based on MPI [75], and a benchmark based on Charm++ [60]. The following systems were used to test the applications:

- Ranger supercomputer
- Taub (an HPC optimized cluster)
- Open Cirrus (physical nodes with commodity Interconnect)
- Private cloud
- Public cloud
- Amazon EC2-CC cloud

They ran the following HPC applications on the above machines:

- Jacobi2D
- NAMD [23]
- ChaNGa [55]
- Sweep3D [128]
- NQueens

The authors made three observations based on running the above HPC application on all the machines discussed above:

1. Some application scaled really well on all platform
   (a) Applications such as Jacobi-2D and NQueens scaled well on all the cores.
2. Scaling only till 32 cores on private cloud
   (a) NAMD and ChaNGa show this behavior. This is the effect of virtualization of the network.
3. Variable runtime for HPC applications on different execution in clouds
   (a) The variability was seen to increase when increasing the scaling.

The authors used various MPE, Jumpshot [131], and Projection tools to trace the HPC applications communication characteristics. On analyzing the communication, the authors have come to a conclusion that communication performance is a major bottleneck. They also conclude that virtualization decreases the performance of an HPC application. This is due to the presence of high latency and reduction in bandwidth. It was also observed that there are random idle times which the authors attribute to interference by other systems. The authors also reach a conclusion on the variability of the running time of HPC application. They say that it is due to the coupling of heterogeneous components in hardware and due to sharing of the virtual machines by external users.

Jackson et al. [53] discuss the performance analysis of HPC applications on the cloud. They compare conventional high-performance computing platforms such as supercomputers to Amazon EC2 cloud using HPC applications. The authors use the NERSC benchmarking framework [29, 32, 49, 75, 96, 101–103, 125] to evaluate the performance of HPC applications on Amazon EC2. In addition to NERSC, they also use integrated performance monitoring (IPM) framework. This framework will provide us with details on the time spent by the application on computation and on networking. They use four machines for this evaluation. The first of such machine is called Carver, which is a four hundred node cluster, which belongs to Lawrence Berkeley National Laboratory. It uses a quad-core Intel Nehalem processor @ 2.67 GHz. Each node has twenty-four GB of RAM. The second machine to be tested is Franklin which is a CrayXT4 supercomputer consisting of 9660 nodes. It has a single AMD Budapest processor @ 2.3GHz. The third system to be tested is Lawrencium which is a Linux cluster that has 198 nodes and the third system is the Amazon EC2 cloud. For testing purposes, they used four compute units of Amazon where one compute unit is equal to 1.2 GHz of Xeon 2007 processors. Normally all of the traditional HPC has a shared parallel file system that between the master nodes and the slave nodes. This is recreated in the cloud environment using virtual clusters [38, 40, 62]. A number of python scripts were used to configure the master node and the slave nodes. The master node would submit the jobs to the slave nodes using MPI and the file system will be shared between the nodes. The shared file system is implemented with the help of elastic block store [13] device which is attached to the virtual machine. The ext3 file system was used in this disk. Eight different HPC applications were evaluated by the authors from the benchmarking suite. These applications are (1) the community atmosphere model, (2) the general atomic and molecular electronic structure system, (3) GTC, (4) IMPACT-T, (5) MAESTRO, (6) MILC, (7) PARATEC, and (8) HPCC. Sustained system performance [68] for each of these applications was computed based on the NERSC benchmark as follows:

$$SSP = N \left(\prod_{i=1}^{M} P_i \right)^{(1/M)} \tag{18.3}$$

where P_i is the performance figure in gigaflops per second per core, N is the number of computational cores. Basically, SSP is the geometric mean of P_i over M applications multiplied by N. It was observed that Lawrencium and Amazon EC2 were the worst performers in terms of computation. Moreover, the network latency is very poor in both of these systems. It was observed that EC2 was 20 times worst performer than the penultimate worst performer. The memory access in the EC2 platform is 10 times slower than the next worst performer Lawrencium. Totally the results indicate that the network has a very high impact on the performance of HPC applications on the cloud.

Roberto et al. [35] analyze the performance of HPC applications on the cloud. They analyze major performance bottlenecks in cloud platform using Amazon EC2 cluster [33] computing environment. Amazon provides two cluster computing

instances named CC1 and CC2. CC1 consists of two quad-core processors whereas CC2 consists of two octa-core processors. They are made keeping in mind the requirement of HPC applications and high network using applications [25]. The authors evaluate 64 instances of CC1 and 32 instances of CC2, which sums up to a total of 512 cores. NAS parallel benchmark suite [19] and NPB multi-zone suite [56] are used for evaluation. Both of these cluster instances use Xen hypervisor for virtualization. The input–output is managed by paravirtual drivers that improve the performance as compared to normal clouds. OpenMPI [92] and MPICH2 [93] are used as the messaging middleware for codes using C/C++. Whereas, for Java-based code they use FastMPJ [120]. Initially, they performed a micro-benchmarking of data transfer from one point to another. They analyzed data transfer between both the inter-cluster and intra-cluster. Micro-benchmarking was conducted using Intel MPI benchmarks suite [126]. Then they analyzed the performance of the HPC kernels especially the effects on scalability due to paravirtualization. This was performed using NAS parallel benchmarks. Then they analyze the suitability of the amazon-based cloud networks for HPC application execution. The metrics they consider are millions of operation per second (MOPS). From the inter-VM communication analysis, it was found that OpenMPI and FastMPJ had lower start-up latency than MPICH2 but still as compared to an application running on barebones hardware these values are not enough. The results in the octa-core cloud were better than the performance of quad-core cloud. Still the major bottleneck is the delay in networking due to the para-virtualized access of the virtual machines of the cloud. In the Intra-VM data transfer, we only use the shared memory and does not use the network infrastructure and hence it was observed that it was better than Inter-VM performance. In this case, the latency was as low as 0.3 and 0.45 μs on both CC1 and CC2. In CC2, it is slightly higher due to the higher clock frequency of CC2. It was observed that cache hierarchy influences the performance of shared memory in both the cluster instances. HPC kernel analysis was performed using Fourier transform, integer sort, and conjugate gradient applications from the NPB kernels. It was observed that the scalability is higher if shared memory was used for data transfer but the moment the application was run on Inter-VM the performance dropped. The analysis reveals that the para-virtualized access of the network interface card by the cloud results in higher start-up latency which limits the scalability of the HPC applications that use intensive communication. The authors conclude that the major bottleneck in running HPC applications in cloud is the communication bottleneck of the cloud due to para-virtualized access of the network interface card by the cloud platform. It has also concluded that CC1 has better scalability than CC2 even though CC2 has higher computational power. If the performance to cost ratio is compared, then CC1 is better than using CC2. The scalability can be increased by running a single process per VM so that shared memory is used instead of network infrastructure. Multiple levels of parallelism have been shown to increase the scalability and performance such as multi-threading with message passing.

Waseem et al. [4] propose a framework for porting scientific applications to between heterogeneous clouds.

18.3.3 Performance Analysis Tools

Burtscher et al. [26] propose a tool for analyzing the performance of HPC applications. It consists of an advanced engine behind a highly usable GUI for bottleneck analysis. In an application, for each procedure, class, and loop it can analyze core, socket, and other bottlenecks. It then provides a brief evaluation and the steps required to remove that specific bottleneck including strategies to optimize the performance of the code. PerfExpert removes the need to have in-depth knowledge about computer architecture to optimize the HPC code. They also present a new metric for measuring the performance called as LCPI, which stands for local cycles per instruction. It is a combination of measurements from performance counters and architectural parameters. Since the local values for each loop are computed, procedure, and class they call it local CPI. For each procedure, loop, and class it calculates the CPI. It also returns the contribution of the following operations to the CPI: (1) memory access data, (2) memory access by instruction, (3) data TLB access, (4) instruction TLB access, (5) FP operations, and (6) branches in loops. It has fifteen performance counters to measure the overall LCPI and the LCPI associated with the six operations discussed above. It also calculates the upper bound of the latency caused due to the six operations discussed above. Some of the major performance counters provided by PerfExpert are L1 and L2 cache access by both data and instruction, total cycles and instruction in the HPC code, L2 cache miss by both data and instruction, instruction and data TLB miss, branch prediction and operations such as floating point addition, subtraction, and multiplication. These LCPI parameters are combined with various system parameters to find important bottlenecks. This can be used to restrict the conceivable causes of bottlenecks. For example, the contribution of a branch statement to the LCPI is given by

$$(BR\_NS * BR\_latency + BR\_MSP * BR\_misslat)/TOT\_INS \qquad (18.4)$$

where $BRINS$ is the total branch instruction, $BRMSP$ indicates the branches missed, and $TOTINS$ is the total instructions in the HPC code. $BRmisslat$ is the miss prediction latency by the CPU and $BRlatency$ is the latency of branch. PerfExpert runs HPCToolkit [30] under its hood. The HPC code is run by the PerfExpert multiple times over the HPCToolkit. It saves the data from various performance counters to a file. The data is then accessed by PerfExpert to find out the bottlenecks in the code. On the basis of the analysis, it will then provide optimization suggestions for the code.

Knupfer et al. [65] discuss a toolset for evaluating the performance of HPC applications called the Vampir toolset. The Vampir toolset mainly consists of three components. They are (1) VampireTrace, (2) a set of visualization tools named Vampir, and (3) Vampir server. VampireTrace is an application tracer for HPC applications. Tracing an HPC application requires Instrumentation which is the modification of the application being traced to detect various events occurring. VampireTrace provides four different kinds of instrumentation, namely compiler

instrumentation, source to source instrumentation, library instrumentation, and manual instrumentation. These modifications are performed at the build time of application. Special flags are provided for the compiler to generate calls for instrumentation. It supports a number of compilers such as GCC, Intel compiler suite, IBM compiler suite, Sun compilers, and NEC compilers. The major disadvantage associated with this technique is the large size of the trace file that is generated. Source to source instrumentation is used by Vampir to instrument the MPI application. Library instrumentation replaces the existing libraries of the HPC application with the libraries required for instrumentation. The major advantage of this technique is that without compiling and link frequently. The disadvantage of this technique is that it requires that all the APIs be replaced by the new libraries. Manual instrumentation is used to get more powerful control on what events need to be traced and which events not. The Vampir toolkit can record the following events.

1. Hardware performance counters: Can find out various performance parameters such as statistics on cache performance, statistics on branch predictions, and statistics on floating point operations.
2. Memory usage of the application: It can trace the memory usage of the HPC application dynamically [59]. It replaces the normal memory functions such as malloc, realloc, and free with special wrappers from the GCC compiler library.
3. Input and output activity tracking: Each and every input and output activity performed by the HPC application can be traced by the VampireTrace by intercepting the I/O calls made by the application.
4. Performance counters defined by the users: The users can define a number of performance counters such as loop counts, results, or other scalar quantities.

Tracing can cause an overhead in the system which results in the performance degradation of the HPC application and might alter the original characteristics of the application. The overhead is introduced mainly at four places in the system, namely the initialization phase, during event handling, during storage of tracing information to disk, and during finalization. Vampir server is a client–server framework that uses distributed systems for the evaluation of HPC application evaluation. We use a parallel production environment as the server and the clients can be desktop computers connected remotely for envisaging the performance graphically. The graphical client enhances the understandability of the system by showing graphical results of the evaluation. It provides various graphical timelines of the application execution. The timelines consist of a global timeline that shows the timeline of all process and threads, a summary timeline that provides the process that is involved with various activity over a period of time. The other two timelines provided are the counter timeline and the process timeline. The counter timeline shows the state of counters with respect to time. Other than timeline display it also provides various statistical displays such as the summary chart, activity chart, message statistics, and input–output statistics. The authors conclude that it is a robust tool that provides a good analysis of HPC applications.

Wolf et al. [127] discuss SCALSCA toolset that has been specially designed to analyze the performance of high-performance parallel applications. It provides a complete runtime summaries and provides insight into the application behavior through various techniques such as tracing and measuring various parameters. It has been designed to be used with large end HPC systems such as IBM blue gene. SCALASCA can identify uneven workloads and evaluate it. It can also detect wait states that occur due to the above-mentioned phenomenon. SCALASCA can be used for applications using OpenMP, MPI, and other hybrid applications written in C/C++ and FORTRAN. It can be run on a wide range of platforms. SALASCA is accessed using the command scales with various flags. Initially, before we start the analysis the code to be analyzed has to be instrumented. By instrumentation, it means that the code has to be modified to record and evaluate various events related to the performance of the system. This process is automated in some platforms whereas in some other platforms it has to be done manually. After this, the instrumented code is run. After the run, there is the option to get the trace of individual runtime events from which a GUI-based timeline representation can be created. Alternatively, a profile that aggregates the performance of various events and a summary report from it can be created. The second option provides an overall summary whereas the trace provides a detailed information. The trace produces a trace file that is used loaded into the memory for evaluation. During the evaluation, it detects various events of significance. It performs a pattern analysis and provides a report about this analysis. Both the pattern report and summary produced contain various performance metrics that is useful for the user. The result of tracing can also be analyzed by other third party tools. Some of the disadvantages of this are that the trace analysis for OpenMP is done serially and the summary produced has only metrics for OpenMP. The authors are improving it by incorporating more functionalities. They are trying to come up with a workaround for the restrictions imposed by the CUBE file format. The bottlenecks or the performance degradation might occur at a later time than the time at which the actual event took place. They are also trying to find a workaround for this issue.

18.3.4 Performance Analysis of Exascale Systems

Abraham et al. [1] discuss the possible performance evaluation of HPC applications on exascale systems. Of all the HPC applications present, only a few HPC applications are capable of exploiting even the petaflop systems [18]. The traditional measurement-based evaluation on exascale system cannot be done because an exascale system does not yet exist. The authors identify the challenges and provide solutions for various problems that are faced when running an HPC application on the exascale system including optimization of the code, formal modeling, and static and runtime analysis. They also propose a conceptual framework for HPC application performance analysis to run it on an exascale system. They try to adapt the HPC application code to achieve good utilization of resources abstractly.

For this, they get the resource usage footprint of various modules at various granularity. An abstract behavioral specification language to describe both the task and deployment model can be used [58]. Also, the language should have big parallel operators over big resource footprint [106]. A resource footprint can be specified using the standard model in model driven development when the code is developed from scratch. The authors also apply formal methods to the code, unlike others who monitor the code and report various statistics regarding the performance. They also provide the challenges faced at runtime analysis such as the absence of a good tool to measure energy metrics with accuracy and given time [9, 24]. Current runtime analysis includes runtime measurement like profiling and tracing data [25]. This has many disadvantages such as manual changing of code and ineffective improvement in performance. They also provide a conceptual framework for designing HPC applications for exascale. The major components in this framework can be summarized as follows:

- Use of domain specific exascale language (DSEL): This is useful for expressing the non-functional aspects of execution.
- Scalable model-based analyzer (SMA): They are responsible for monitoring and evaluating resource consumption.
- Exascale runtime data collector (ERDC): It performs functionalities such as data mining, filtering, and data analysis.
- Autonomous feedback loop (AFL): AFL gets the feedback from the runtime that is fed back into the model to tune the model.

Energy-Efficient Systems (Embedded Systems Nodes)

One of the major roadblocks for the adoption of exascale systems for HPC computations is the energy consumption of exascale systems. A supercomputer is not supposed to exceed 20 MW. We would require an efficiency of almost fifty gigaflops (GFLOPS) per watt to build an exascale system below 20 MW. To break through this barrier, we need to increase the efficiency of the current systems by a factor of twenty-five, whereas in the embedded systems the energy is of great importance. Therefore, one way to do this is to use the components used in the embedded systems as analyzed by Stanisic et al. [115]. Stanisic et al. [115] developed HPC clusters with the help of low-power embedded components and they evaluate various existing HPC applications on this embedded platform. They study the scalability of the existing HPC applications to the low-power embedded system-based HPC clusters. They used eleven HPC applications to run on the low-power HPC clusters. SPECFEM3D and BigDFT are the two major HPC applications used by them. The authors have developed a board called snowboard which is a fully embedded computer by itself. It uses a powerful ARM dual cortex A9 running at 1GHz. It has 8 GB of e-MMC memory. It has a full HD supported HDMI port along with Mali 400 GPU. The power consumption of this board is just 2.5 W. This board is compared with an Intel Xeon running at quad-core CPU at 2.6 GHz and has

a power consumption of 95 W. Three benchmarks are used to compare these two systems: (1)LINPACK, (2) CoreMark, and (3) StockFish. It was found that it takes the same amount of energy on both platforms to run LINPACK but benchmarking SPECFEM3D using CoreMark requires only less than 5 times of energy required by Xeon. It was found out that the applications scale well for more than 90% on the embedded low-power platform. The scalability is almost ideal when using SPECFEM3D whereas BigDFT showed a lesser scalability. They further investigate the performance of HPC application by analyzing the memory. For this, they run a heavy memory rigorous kernel as shown in [121]. In this technique they measure the time it takes to access the elements of a fixed size array inside a loop using fixed steps. The memory bandwidth is calculated after running the above kernel. It is the ratio of the total number of access that was required to access to the time it took to execute the kernel. The memory structure of ARM is dissimilar to the Intel architecture. Hence, different behaviors were observed. A large number of cache miss was observed because for an array of size 32 KB the page allocation was not consecutive. Moreover, the memory had to be frequently cleaned. The use of real-time schedulers also resulted in the degradation of the HPC applications performance which is caused by wrong decisions taken by the OS scheduler. A variety of code optimization was done to HPC applications including changing element size, unrolling of loop, etc. Increasing the size of variables as well as loop rolling had optimistic results. There was quite an improvement in the performance due to the doubling of the bandwidth. The authors conclude that more HPC code optimization has to be performed to run HPC applications on low-power HPC platforms.

18.4 Discussion and Analysis

In this section we shall provide a qualitative analysis and discuss the performance analysis tools. Table 18.3 summarizes some of the major performance analysis tools that have been reviewed in this paper.

Most of the researchers have run the HPC applications on the cloud to study the feasibility of HPC applications on the cloud. References [21, 35, 47, 53, 120] ran various HPC applications on the cloud. One of the major findings of all these studies was that cloud as a platform does not scale well for HPC applications. One of the major reasons for this non-scalability after a certain extent was due to the network communication between the virtual machines. It scaled really well if the application was run using a single VM. The reason for this is due to the virtualization of the network interface card leads to degradation in the speed of the network. Shared memory can be used to reduce the network communication to increase the performance of the cloud-based application. The performance to cost ratio of the clouds is also that enticing even though some packages offered do provide the good cost to performance ratio as compared to other HPC platforms such as grids, clusters, and supercomputers. From the discussion of various researchers, we

Table 18.3 A summary and comparison of various performance analysis tools for HPC applications

Features	Vampir Suite	PerfExpert	Scalasca	Periscope	Kojak	HPCToolkit
Auto-instrumentation	Yes	Yes	Yes	Yes	Yes	Yes
Bottlenecks	Yes	Yes	Yes	Yes	Yes	Yes
Data visualization	Yes	No	Yes	No	No	Yes
GUI	Yes	No	Yes	Yes	No	Yes
Language	C++	C++/C	C++/C	C++/C	C/F90	C++
Open source	No	Yes	No	No	No	Yes
Profiling	Yes	Yes	Yes	No	No	Yes
Restructure	No	Yes	Yes	No	No	No
Tracing	Yes	Yes	Yes	Yes	Yes	Yes
Timeline	Yes	No	Yes	No	No	No

conclude that the scalability of HPC applications on the cloud depends on a number of factors such as network communication especially between the virtual and real components. Combining message passing with multi-threading also increases the scalability of the HPC applications on the cloud platform. HPC applications with less communication and less interference sensitivity are suitable for the cloud. The HPC applications have to be modified to make it cloud-aware.

Moreover, we need to consider the future wherein we have to efficiently use the exascale systems. A major roadblock for this is the power consumption and hence we found that HPC applications can also be executed on low cost embedded hardware [115]. The MONT-BLANC project deals with analyzing the performance of HPC applications on low-power embedded systems. It was observed that it performs really well on embedded platform but had issues with real-time scheduling and physical page application. A framework for exascale systems has been developed by Ábrahám et al. [1]. This will be useful for modifying the HPC application accordingly and in increasing their performance. Graphical processing units (GPUs) are also among the highest power-efficient devices. A large number of supercomputers in the Top500 list[1] use GPUs. The Green500[2] lists the top 500 supercomputers in the world by energy efficiency. Supercomputers with the highest Flops/Watts ratio are ranked first in this list. GPUs are the accelerating unit for seven supercomputers in the top ten in Green500.[3]

A number of performance analysis tools exist that help in analyzing the performance of the HPC applications. A brief summary and comparison are provided in Table 18.3. All of the tools provide profiling and tracing features. Profiling is the feature where it would provide various global performance metrics. Tracing consists of following the code in runtime and tracing the code. These actually require the

[1] https://www.top500.org/.

[2] https://www.top500.org/green500/.

[3] https://www.top500.org/green500/lists/2018/06/.

modification of the code. This process of modification is called instrumentation. All of them also provide instrumentation support at various levels. They help in detecting the bottlenecks, but all of them do not give suggestion and where to change the code and what code needs to be changed. An example of this is PerfExpert that provides the complete details on code creating the bottleneck and the restructuring required. But PerfExpert does not provide GUI support whereas others provide a complete statistical information using charts, graphs, and timelines in which you can zoom through various scales of timescale. The Vampir toolkit provides separate timelines for the summary, counter, and various process. It also provides statistical charts for various activities, message statistics, and I/O activities. SCALSCA is also very similar to PerfExpert, but PerfExpert is open source whereas SCALASCA is proprietary software.

18.5 Future Research and Issues

There are a lot of issues associated with analyzing the performance of HPC application. We list some of the major issues and the direction of the required future work.

- Most of the tools available are for the Linux platform. Hence, we need to port various performance analysis tools to Windows platform
- There are a lot of redundancies when the application performs a trace. Eliminating redundancy in trace has to be done.
- Pattern detection in large datasets using complete call graphs [37].
- Applications that use single instruction multiple data (SIMD) consists of redundancies in execution. It is required to replace these redundant behaviors with a single instance which will result in higher performance and lesser memory.
- More case studies with more applications on more platforms are required.
- Expand current capabilities of the performance analysis tools by using counters that are not based on performance.
- The performance analysis tools should automatically not only suggest solutions for the bottlenecks but also automatically implement these into the code for all kinds of bottlenecks.
- The performance analysis tools should also provide a higher level of input–output optimization.
- The variability and the reduction of speed in clouds were attributed to the network communication in the cloud. This issue of network delay has to be mitigated.
- Further improvement in the scalability of the performance analysis tools is required.
- The OpenMP analysis in SCALASCA performance analysis tool has to be parallelized.
- There are no existing tools to analyze the input–output in MPI, currently this is performed by serial tracer in SCALASCA.

– The current file formats that are used for the performance analysis provide a lot of restrictions such as storing and processing of metrics that cannot be aggregated. An efficient file system has to be developed.
– The bottlenecks that appear in the application might appear at a later time than when the event that caused it occurs. The performance analysis tools should be able to figure out the events that caused the issue in such occurrence.
– HPC applications have to be modified so that it performs efficiently in exascale and petascale.

18.6 Conclusions

In this chapter, we reviewed the works on analyzing HPC applications and discussed several performance analysis tools for HPC applications. The researchers observed the performance of the HPC applications on various platforms and identified various bottlenecks and issues related to the performance of the HPC application on parallel platforms. Some researchers have discussed the usage of performance tools for binding and replacing the bottleneck in the HPC applications. We also studied and qualitatively compared various tools for performance analysis of HPC applications. We discussed the performance of various HPC applications on a number of HPC platforms. Moreover, we discussed various HPC benchmarking suites and various HPC applications being used for the performance analysis and thereafter, we also discussed the future research directions and issues.

Acknowledgements The authors acknowledge with thanks the technical and financial support from the Deanship of Scientific Research (DSR) at the King Abdulaziz University (KAU), Jeddah, Saudi Arabia, under the grant number G-651-611-38. The work carried out in this paper is supported by the High Performance Computing Center at the King Abdulaziz University, Jeddah.

References

1. Ábrahám, E., Bekas, C., Brandic, I., Genaim, S., Johnsen, E.B., Kondov, I., Pllana, S., Streit, A.: Preparing HPC applications for exascale: Challenges and recommendations (2015). CoRR abs/1503.06974. http://arxiv.org/abs/1503.06974
2. Abraham, M.J., Murtola, T., Schulz, R., Páll, S., Smith, J.C., Hess, B., Lindahl, E.: Gromacs: high performance molecular simulations through multi-level parallelism from laptops to supercomputers. SoftwareX **1-2**, 19–25 (2015). http://www.sciencedirect.com/science/article/pii/S2352711015000059
3. Agullo, E., Demmel, J., Dongarra, J., Hadri, B., Kurzak, J., Langou, J., Ltaief, H., Luszczek, P., Tomov, S.: Numerical linear algebra on emerging architectures: the plasma and magma projects. J. Phys. Conf. Ser. **180**, 012037 (2009)
4. Ahmed, W., Khan, M., Khan, A.A., Mehmood, R., Algarni, A., Albeshri, A., Katib, I.: A framework for faster porting of scientific applications between heterogeneous clouds. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. pp. 27–43. Springer International Publishing, Cham (2018)

5. Alam, F., Mehmood, R., Katib, I., Albogami, N.N., Albeshri, A.: Data fusion and IoT for smart ubiquitous environments: a survey. IEEE Access **5**, 9533–9554 (2017)
6. Alam, F., Mehmood, R., Katib, I.: D2TFRS: an object recognition method for autonomous vehicles based on RGB and spatial values of pixels. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. pp. 155–168. Springer International Publishing, Cham (2018)
7. Alamoudi, E., Mehmood, R., Albeshri, A., Gojobori, T.: Dna profiling methods and tools: a review. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 216–231. Springer International Publishing, Cham (2018)
8. Alomari, E., Mehmood, R.: Analysis of tweets in Arabic language for detection of road traffic conditions. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. pp. 98–110. Springer International Publishing, Cham (2018)
9. Alonso, P., Badia, R.M., Labarta, J., Barreda, M., Dolz, M.F., Mayo, R., Quintana-Orti, E.S., Reyes, R.: Tools for power-energy modelling and analysis of parallel scientific applications. In: 2012 41st International Conference on Parallel Processing (ICPP), pp. 420–429. IEEE, New York (2012)
10. Alotaibi, S., Mehmood, R.: Big data enabled healthcare supply chain management: opportunities and challenges. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 207–215. Springer International Publishing, Cham (2018)
11. Alyahya, H., Mehmood, R., Katib, I.: Parallel sparse matrix vector multiplication on Intel MIC: performance analysis. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 306–322. Springer International Publishing, Cham (2018)
12. Alzahrani, S., Ikbal, M.R., Mehmood, R., Fayez, M., Katib, I.: Performance evaluation of Jacobi iterative solution for sparse linear equation system on multicore and manycore architectures. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 296–305. Springer International Publishing, Cham (2018)
13. Amazon: AWS | Amazon Elastic Block Store (EBS) - Incremental Backup & Persistent Storage. http://aws.amazon.com/ebs/
14. Aqib, M., Mehmood, R., Albeshri, A., Alzahrani, A.: Disaster management in smart cities by forecasting traffic plan using deep learning and GPUs. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 139–154. Springer International Publishing, Cham (2018)
15. Arfat, Y., Aqib, M., Mehmood, R., Albeshri, A., Katib, I., Albogami, N., Alzahrani, A.: Enabling smarter societies through mobile big data fogs and clouds. Proc. Comput. Sci. **109**, 1128–1133 (2017). http://www.sciencedirect.com/science/article/pii/S1877050917311213. 8th International Conference on Ambient Systems, Networks and Technologies, ANT-2017 and the 7th International Conference on Sustainable Energy Information Technology, SEIT 2017, 16-19 May 2017, Madeira
16. Arfat, Y., Mehmood, R., Albeshri, A.: Parallel shortest path graph computations of United States road network data on apache spark. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 323–336. Springer International Publishing, Cham (2018)
17. Azad, A., Ballard, G., Buluç, A., Demmel, J., Grigori, L., Schwartz, O., Toledo, S., Williams, S.: Exploiting multiple levels of parallelism in sparse matrix-matrix multiplication. SIAM J. Sci. Comput. **38**(6), C624–C651 (2016). https://doi.org/10.1137/15M104253X
18. Bader, D.A.: Petascale Computing: Algorithms and Applications. CRC Press, Boca Raton (2007)
19. Bailey, D.H., Barszcz, E., Barton, J.T., Browning, D.S., Carter, R.L., Dagum, L., Fatoohi, R.A., Frederickson, P.O., Lasinski, T.A., Schreiber, R.S., et al.: The NAS parallel benchmarks. Int. J. High Perform. Comput. Appl. **5**(3), 63–73 (1991)

20. Bailey, J.A., Bazavov, A., Bernard, C., Bouchard, C.M., DeTar, C., Du, D., El-Khadra, A.X., Foley, J., Freeland, E.D., Gámiz, E., Gottlieb, S., Heller, U.M., Kim, J., Kronfeld, A.S., Laiho, J., Levkova, L., Mackenzie, P.B., Meurice, Y., Neil, E.T., Oktay, M.B., Qiu, S.W., Simone, J.N., Sugar, R., Toussaint, D., Van de Water, R.S., Zhou, R.: Refining new-physics searches in $b \to d\tau\nu$ with lattice QCD. Phys. Rev. Lett. **109**, 071802 (2012). https://link.aps.org/doi/10.1103/PhysRevLett.109.071802
21. Benedict, S.: Performance issues and performance analysis tools for HPC cloud applications: a survey. Computing **95**(2), 89–108 (2013)
22. Berriman, G.B., Juve, G., Deelman, E., Regelson, M., Plavchan, P.: The application of cloud computing to astronomy: A study of cost and performance. In: 2010 Sixth IEEE International Conference on e-Science Workshops, December, pp. 1–7 (2010)
23. Bhatele, A., Kumar, S., Mei, C., Phillips, J.C., Zheng, G., Kale, L.V.: Overcoming scaling challenges in biomolecular simulations across multiple platforms. In: IEEE International Symposium on Parallel and Distributed Processing, 2008 (IPDPS 2008), pp. 1–12. IEEE, New York (2008)
24. Bohra, A.E.H., Chaudhary, V.: Vmeter: power modelling for virtualized clouds. In: 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and PhD Forum (IPDPSW), pp. 1–8. IEEE, New York (2010)
25. BPG: Best Practice Guides. http://www.prace-ri.eu/best-practice-guides/
26. Burtscher, M., Kim, B.D., Diamond, J., McCalpin, J., Koesterke, L., Browne, J.: PerfExpert: an easy-to-use performance diagnosis tool for HPC applications. In: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–11. IEEE Computer Society, Washington (2010)
27. Carrington, L.C., Laurenzano, M., Snavely, A., Campbell Jr., R.L., Davis, L.P.: How well can simple metrics represent the performance of HPC applications? In: Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference, pp. 48–48. IEEE, New York (2005)
28. Carrington, L., Snavely, A., Wolter, N.: A performance prediction framework for scientific applications. Fut. Gener. Comput. Syst. **22**(3), 336–346 (2006)
29. Carter, J., Oliker, L., Shalf, J.: Performance evaluation of scientific applications on modern parallel vector systems. In: High Performance Computing for Computational Science-VECPAR 2006, pp. 490–503. Springer, New York (2007)
30. Djoudi, L., Barthou, D., Carribault, P., Lemuet, C., Acquaviva, J.T., Jalby, W.: Exploring application performance: a new tool for a static/dynamic approach. In: Proceedings of the 6th LACSI Symposium (2005)
31. Dongarra, J.L.A.P.: The LINPACK benchmark: past, present and future. Concurr. Comput. Pract. and Exp. **15**, 1–18 (2003)
32. Dunigan Jr, T.H., Vetter, J.S., White III, J.B., Worley, P.H.: Performance evaluation of the Cray x1 distributed shared-memory architecture. Micro, IEEE **25**(1), 30–40 (2005)
33. ECC2. Elastic Compute Cloud (EC2) Cloud Server & Hosting – AWS. //aws.amazon.com/ec2/
34. Eleliemy, A., Fayez, M., Mehmood, R., Katib, I., Aljohani, N.: Loadbalancing on parallel heterogeneous architectures: Spin-image algorithm on CPU and MIC. In: 9th EUROSIM Congress on Modelling and Simulation. EUROSIM (2016). http://edoc.unibas.ch/53117/
35. ExpóSito, R.R., Taboada, G.L., Ramos, S., Touriño, J., Doallo, R.: Performance analysis of HPC applications in the cloud. Fut. Gener. Comput. Syst. **29**(1), 218–229 (2013)
36. Farber, R.: The convergence of big data and extreme-scale HPC (2018). https://www.hpcwire.com/2018/08/31/the-convergence-of-big-data-and-extreme-scale-hpc/
37. Ferreira, G., Kästner, C., Pfeffer, J., Apel, S.: Characterizing complexity of highly-configurable systems with variational call graphs: analyzing configuration options interactions complexity in function calls. In: Proceedings of the 2015 Symposium and Bootcamp on the Science of Security. p. 17. ACM, New York (2015)
38. Foster, I., Freeman, T., Keahy, K., Scheftner, D., Sotomayer, B., Zhang, X.: Virtual clusters for grid communities. In: Sixth IEEE International Symposium on Cluster Computing and the Grid, 2006 (CCGRID 06), vol. 1, pp. 513–520. IEEE, New York (2006)

39. Freche, J., Frings, W., Sutmann, G.: High-throughput parallel-I/O using SIONlib for mesoscopic particle dynamics simulations on massively parallel computers. In: Parallel Computing: From Multicores and GPU's to Petascale Advances in Parallel Computing, vol. 19, pp. 371–378. IOS Press, Amsterdam (2010)
40. Freeman, T., Keahey, K., Sotomayor, B., Zhang, X., Foster, I., Scheftner, D.: Virtual clusters for grid communities. Citeseer (2006)
41. Gel, A., Hu, J., Ould-Ahmed-Vall, E., Kalinkin, A.A.: Modernization and optimization of a legacy open-source CFD code for high-performance computing architectures. Int. J. Comput. Fluid Dynam. **31**(2), 122–133 (2017). https://doi.org/10.1080/10618562.2017.1285398
42. Genovese, L., Videau, B., Ospici, M., Deutsch, T., Goedecker, S., Méhaut, J.F.: Daubechies wavelets for high performance electronic structure calculations: The BigDFT project. Comptes Rendus Mécanique **339**(2), 149–164 (2011). http://www.sciencedirect.com/science/article/pii/S1631072110002135. High Performance Computing
43. Giannozzi, P., Baroni, S., Bonini, N., Calandra, M., Car, R., Cavazzoni, C., Ceresoli, D., Chiarotti, G.L., Cococcioni, M., Dabo, I., et al.: Quantum espresso: a modular and open-source software project for quantum simulations of materials. J. Phys. Condens. matter **21**(39), 395502 (2009)
44. Gibbon, P.: Pepc: pretty efficient parallel coulomb-solver. Sonstiger Interner Bericht ZAM-IB-2003-05, ZAM, Jülich, Forschungszentrum (2003)
45. Gordon, M.S., Schmidt, M.W.: Advances in electronic structure theory: GAMESS a decade later. In: Dykstra, C.E., Frenking, G., Kim, K.S., Scuseria, G.E. (eds.) Theory and Applications of Computational Chemistry, chapter 41, pp. 1167–1189. Elsevier, Amsterdam (2005). http://www.sciencedirect.com/science/article/pii/B9780444517197500846
46. Gudiksen, B.V., Carlsson, M., Hansteen, V.H., Hayek, W., Leenaarts, J., Martínez-Sykora, J.: The stellar atmosphere simulation code Bifrost - code description and validation. Astron. Astrophys. **531**, A154 (2011). https://doi.org/10.1051/0004-6361/201116520
47. Gupta, A., Faraboschi, P., Gioachin, F., Kale, L., Kaufmann, R., Lee, B.S., March, V., Milojicic, D., Suen, C.: Evaluating and improving the performance and scheduling of HPC applications in cloud. IEEE Trans. Cloud Comput. **4**(99), 1–1 (2014)
48. Gustafson, J.L., Todi, R.: Conventional benchmarks as a sample of the performance spectrum. In: Proceedings of the Thirty-First Hawaii International Conference on System Sciences, 1998, vol. 7, pp. 514–523. IEEE, New York (1998)
49. Gygi, F., Yates, R.K., Lorenz, J., Draeger, E.W., Franchetti, F., Ueberhuber, C.W., Supinski, B.R.D., Kral, S., Gunnels, J.A., Sexton, J.C.: Large-scale first-principles molecular dynamics simulations on the Bluegene/l platform using the Qbox code. In: Proceedings of the 2005 ACM/IEEE conference on Supercomputing, p. 24. IEEE Computer Society, Washington (2005)
50. Heck, D., Pierog, T., Knapp, J.: CORSIKA: An Air Shower Simulation Program. Astrophysics Source Code Library (2012)
51. Hwu, W.M., Chang, L.W., Kim, H.S., Dakkak, A., El Hajj, I.: Transitioning HPC software to exascale heterogeneous computing. In: Computational Electromagnetics International Workshop (CEM), July 2015, pp. 1–2 (2015)
52. Irbäck, A., Mohanty, S.: Profasi: A Monte Carlo simulation package for protein folding and aggregation. J. Comput. Chem. **27**(13), 1548–1555. https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.20452
53. Jackson, K.R., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., Wasserman, H.J., Wright, N.J.: Performance analysis of high performance computing applications on the amazon web services cloud. In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), pp. 159–168. IEEE, New York (2010)
54. Jacobsen, N.G., Fuhrman, D.R., Fredsøe, J.: A wave generation toolbox for the open-source CFD library: Openfoam®. Int. J. Numer. Methods Fluids **70**(9), 1073–1088. https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.2726
55. Jetley, P., Gioachin, F., Mendes, C., Kale, L.V., Quinn, T.: Massively parallel cosmological simulations with ChaNGa. In: International Symposium on Parallel and Distributed Processing, 2008 (IPDPS 2008), pp. 1–12. IEEE, New York (2008)

56. Jin, H., Van der Wijngaart, R.F.: Performance characteristics of the multi-zone NAS parallel benchmarks. In: Proceedings of the 18th International Parallel and Distributed Processing Symposium, 2004, p. 6. IEEE, New York (2004)
57. Jöckel, P., Sander, R., Kerkweg, A., Tost, H., Lelieveld, J.: Technical note: the modular earth submodel system (MESSy) - a new approach towards earth system modeling. Atmos. Chem. Phys. **5**(2), 433–444 (2005). https://www.atmos-chem-phys.net/5/433/2005/
58. Johnsen, E.B., Hähnle, R., Schäfer, J., Schlatte, R., Steffen, M.: ABS: a core language for abstract behavioral specification. In: Formal Methods for Components and Objects, pp. 142–164. Springer, New York (2012)
59. Jurenz, M., Brendel, R., Knüpfer, A., Müller, M., Nagel, W.E.: Memory allocation tracing with VampireTrace. In: Computational Science–ICCS 2007, pp. 839–846. Springer, New York (2007)
60. Kale, L.V., Krishnan, S.: CHARM++: A Portable Concurrent Object Oriented System Based on C++, vol. 28. ACM, New York (1993)
61. Kay, J.E., Deser, C., Phillips, A., Mai, A., Hannay, C., Strand, G., Arblaster, J.M., Bates, S.C., Danabasoglu, G., Edwards, J., Holland, M., Kushner, P., Lamarque, J.F., Lawrence, D., Lindsay, K., Middleton, A., Munoz, E., Neale, R., Oleson, K., Polvani, L., Vertenstein, M.: The community earth system model (CESM) large ensemble project: a community resource for studying climate change in the presence of internal climate variability. Bull. Am. Meteorol. Soc. **96**(8), 1333–1349 (2015). https://doi.org/10.1175/BAMS-D-13-00255.1
62. Keahey, K., Figueiredo, R., Fortes, J., Freeman, T., Tsugawa, M.: Science clouds: early experiences in cloud computing for scientific applications. Cloud Comput. Appl. **2008**, 825–830 (2008)
63. Khanum, A., Alvi, A., Mehmood, R.: Towards a semantically enriched computational intelligence (SECI) framework for smart farming. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. pp. 247–257. Springer International Publishing, Cham (2018)
64. Kirk, B.S., Peterson, J.W., Stogner, R.H., Carey, G.F.: libMesh: a C++ library for parallel adaptive mesh refinement/coarsening simulations. Eng. Comput. **22**(3), 237–254 (2006). https://doi.org/10.1007/s00366-006-0049-3
65. Knüpfer, A., Brunst, H., Doleschal, J., Jurenz, M., Lieber, M., Mickler, H., Müller, M.S., Nagel, W.E.: The Vampir performance analysis tool-set. In: Tools for High Performance Computing, pp. 139–155. Springer, New York (2008)
66. Kodiyalam, S., Yang, R., Gu, L., Tho, C.H.: Multidisciplinary design optimization of a vehicle system in a scalable, high performance computing environment. Struct. Multidiscip. Optim. **26**(3), 256–263 (2004). https://doi.org/10.1007/s00158-003-0343-2
67. Komatitsch, D., Tromp, J.: Introduction to the spectral element method for three-dimensional seismic wave propagation. Geophys. J. Int. **139**(3), 806–822 (1999). https://onlinelibrary.wiley.com/doi/abs/10.1046/j.1365-246x.1999.00967.x
68. Kramer, W., Shalf, J., Strohmaier, E.: The NERSC Sustained System Performance (SSP) Metric. Lawrence Berkeley National Laboratory (2005)
69. Kwiatkowska, M., Mehmood, R.: Out-of-core solution of large linear systems of equations arising from stochastic modelling. In: Hermanns, H., Segala, R. (eds.) Process Algebra and Probabilistic Methods: Performance Modeling and Verification, pp. 135–151. Springer, Berlin/Heidelberg (2002)
70. Kwiatkowska, M., Mehmood, R., Norman, G., Parker, D.: A symbolic out-of-core solution method for Markov models. Electron. Notes Theor. Comput. Sci. **68**(4), 589–604 (2002). http://www.sciencedirect.com/science/article/pii/S1571066105803949
71. Kwiatkowska, M., Parker, D., Zhang, Y., Mehmood, R.: Dual-processor parallelisation of symbolic probabilistic model checking. In: Proceedings of the IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, MASCOTS '04, pp. 123–130. IEEE Computer Society, Washington (2004). http://dl.acm.org/citation.cfm?id=1032659.1034195

72. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) Proceedings of the 23rd International Conference on Computer Aided Verification (CAV'11). Lecture Notes in Computer Science, vol. 6806, pp. 585–591. Springer, New York (2011)
73. Letherwood, M.D., Gunter, D.D.: Ground vehicle modeling and simulation of military vehicles using high performance computing. Parallel Comput. **27**(1), 109–140 (2001). http://www.sciencedirect.com/science/article/pii/S0167819100000910. New Trends in High Performance Computing
74. Lingerfelt, E., Endeve, E., Hui, Y., Smith, C., Somnath, S., Grodowitz, N., Borreguero, J., Bao, F., Niedziela, J., Bansal, D., Delaire, O., Archibald, R., Belianinov, A., Shankar, M., Jesse, S.: BEAM: an HPC pipeline for nanoscale materials analysis and neutron data modeling. In: APS March Meeting Abstracts, p. A7.002 (2017)
75. Lusk, E., Huss, S., Saphir, B., Snir, M.: MPI: a message-passing interface standard (2009)
76. Luszczek, P.R., Bailey, D.H., Dongarra, J.J., Kepner, J., Lucas, R.F., Rabenseifner, R., Takahashi, D.: The HPC challenge (HPCC) benchmark suite. In: Proceedings of the 2006 ACM/IEEE conference on Supercomputing, p. 213. Citeseer (2006)
77. Mantripragada, K., Binotto, A., Tizzei, L., Netto, M.: A feasibility study of using HPC cloud environment for seismic exploration. In: 77th EAGE Conference and Exhibition 2015 (2015)
78. McCalpin, J.D.: Memory bandwidth and machine balance in current high performance computers (1995)
79. Mehmood, R.: A survey of out-of-core analysis techniques in stochastic modelling. Report CSR-03-7, University of Birmingham (2003). https://www.researchgate.net/publication/326827715_A_Survey_of_Out-of-Core_Analysis_Techniques_in_Stochastic_Modelling
80. Mehmood, R.: Disk-based Techniques for Efficient Solution of Large Markov Chains. Thesis (2004)
81. Mehmood, R.: Serial Disk-Based Analysis of Large Stochastic Models, pp. 230–255. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24611-4_7
82. Mehmood, R., Crowcroft, J.: Parallel iterative solution method for large sparse linear equation systems. UCAM-CL-TR-650. Report UCAM-CL-TR-650, University of Cambridge, Computer Laboratory (2005). http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-650.pdf
83. Mehmood, R., Graham, G.: Big data logistics: a health-care transport capacity sharing model. Proc. Comput. Sci. **64**, 1107–1114 (2015). http://www.sciencedirect.com/science/article/pii/S1877050915027015. Conference on ENTERprise Information Systems/International Conference on Project MANagement/Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN/HCist 2015 October 7-9, 2015
84. Mehmood, R., Lu, J.A.: Computational Markovian analysis of large systems. J. Manuf. Technol. Manage. **22**(6), 804–817 (2011). https://doi.org/10.1108/17410381111149657
85. Mehmood, R., Parker, D., Kwiatkowska, M.: An efficient BDD-based implementation of Gauss-Seidel for CTMC analysis. Report CSR-03-13, University of Birmingham (2003). http://www.prismmodelchecker.org/bibitem.php?key=MPK03b
86. Mehmood, R., Crowcroft, J., Elmirghani, J.M.H.: A parallel implicit method for the steady-state solution of CTMCs. In: 14th IEEE International Symposium on Modeling, Analysis, and Simulation, pp. 293–302 (2006)
87. Mehmood, R., Faisal, M.A., Altowaijri, S.: Future networked healthcare systems: a review and case study. In: Boucadair, M., Jacquenet, C. (eds.) Handbook of Research on Redesigning the Future of Internet Architectures, pp. 531–558. IGI Global, Hershey, PA (2015). http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-4666-8371-6.ch022
88. Mehmood, R., Alam, F., Albogami, N.N., Katib, I., Albeshri, A., Altowaijri, S.M.: UTiLearn: a personalised ubiquitous teaching and learning system for smart societies. IEEE Access **5**, 2615–2635 (2017)
89. Mehmood, R., Meriton, R., Graham, G., Hennelly, P., Kumar, M.: Exploring the influence of big data on city transport operations: a Markovian approach. Int. J. Oper. Prod. Manage. **37**(1), 75–104 (2017). https://doi.org/10.1108/IJOPM-03-2015-0179

90. Meinke, J.H., Mohanty, S., Eisenmenger, F., Hansmann, U.H.E.: SMMP v. 3.0-simulating proteins and protein interactions in Python and Fortran. Comput. Phys. Commun. **178**, 459–470 (2008)
91. Moureau, V., Domingo, P., Vervisch, L.: Design of a massively parallel CFD code for complex geometries. Comptes Rendus Mécanique **339**(2), 141–148 (2011). http://www.sciencedirect.com/science/article/pii/S1631072110002111. High Performance Computing
92. MPI: Open MPI: Open Source High Performance Computing. http://www.open-mpi.org/
93. MPICH: MPICH | High-Performance Portable MPI. http://www.mpich.org/
94. Muhammed, T., Mehmood, R., Albeshri, A.: Enabling reliable and resilient IoT based smart city applications. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 169–184. Springer International Publishing, Cham (2018)
95. Muhammed, T., Mehmood, R., Albeshri, A., Katib, I.: Ubehealth: a personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities. IEEE Access **6**, 32258–32285 (2018)
96. Nakajima, K.: Three-level hybrid vs. flat MPI on the earth simulator: parallel iterative solvers for finite-element method. Appl. Numer. Math. **54**(2), 237–255 (2005)
97. NAS: NAS Parallel Benchmarks. http://www.nas.nasa.gov/publications/npb.html
98. Nielsen, E.J., Diskin, B.: High-performance aerodynamic computations for aerospace applications. Parall. Comput. **64**, 20–32 (2017). http://www.sciencedirect.com/science/article/pii/S0167819117300182. High-End Computing for Next-Generation Scientific Discovery
99. Niethammer, C., Gracia, J., Knüpfer, A., Resch, M.M., Nagel, W.E.: Tools for High Performance Computing 2014: Proceedings of the 8th International Workshop on Parallel Tools for High Performance Computing, October 2014, HLRS, Stuttgart. Springer, New York (2015)
100. Nonaka, A., Almgren, A.S., Bell, J.B., Lijewski, M.J., Malone, C.M., Zingale, M.: Maestro: an adaptive low Mach number hydrodynamics algorithm for Stellar flows **188**(2), 358–383 (2010). http://dx.doi.org/10.1088/0067-0049/188/2/358
101. Oliker, L., Canning, A., Carter, J., Shalf, J., Ethier, S.: Scientific computations on modern parallel vector systems. In: Proceedings of the 2004 ACM/IEEE Conference on Supercomputing, p. 10. IEEE Computer Society, Washington (2004)
102. Oliker, L., Carter, J., Wehner, M., Canning, A., Ethier, S., Mirin, A., Parks, D., Worley, P., Kitawaki, S., Tsuda, Y.: Leading computational methods on scalar and vector HEC platforms. In: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing, p. 62. IEEE Computer Society, Washington (2005)
103. Oliker, L., Canning, A., Carter, J., Iancu, C., Lijewski, M., Kamil, S., Shalf, J., Shan, H., Strohmaier, E., Ethier, S., et al.: Scientific application performance on candidate petascale platforms. In: IEEE International Parallel and Distributed Processing Symposium, 2007 (IPDPS 2007), pp. 1–12. IEEE, New York (2007)
104. Pfrommer, B., Raczkowski, D., Canning, A., Louie, S.: Paratec (parallel total energy code), Lawrence Berkeley national laboratory (with contributions from F. Mauri, M. Cote, Y. Yoon, C. Pickard and P. Haynes). www.nersc.gov/projects/paratec
105. Pérez, F.E.H., Mukhadiyev, N., Xu, X., Sow, A., Lee, B.J., Sankaran, R., Im, H.G.: Direct numerical simulations of reacting flows with detailed chemistry using many-core/GPU acceleration. Comput. Fluids **173**, 73–79 (2018). http://www.sciencedirect.com/science/article/pii/S0045793018301786
106. Pllana, S., Brandic, I., Benkner, S.: A survey of the state of the art in performance modeling and prediction of parallel and distributed computing systems. Int. J. Comput. Intel. Res.(IJCIR) **4**, 17–26 (2008)
107. Qiang, J., Lidia, S., Ryne, R.D., Limborg-Deprey, C.: Three-dimensional quasistatic model for high brightness beam dynamics simulation. Phys. Rev. ST Accel. Beams **9**, 044204 (2006). https://link.aps.org/doi/10.1103/PhysRevSTAB.9.044204
108. Reed, D.A., Dongarra, J.: Exascale computing and big data. Commun. ACM **58**(7), 56–68 (2015). http://doi.acm.org/10.1145/2699414

109. Rudi, J., Malossi, A.C.I., Isaac, T., Stadler, G., Gurnis, M., Staar, P.W.J., Ineichen, Y., Bekas, C., Curioni, A., Ghattas, O.: An extreme-scale implicit solver for complex PDEs: highly heterogeneous flow in earth's mantle. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '15, pp. 5:1–5:12. ACM, New York, (2015). http://doi.acm.org/10.1145/2807591.2807675
110. Sáez, X., Soba, A., Sánchez, E., Kleiber, R., Castejón, F., Cela, J.M.: Improvements of the particle-in-cell code EUTERPE for petascaling machines. Comput. Phys. Commun. **182**(9), 2047–2051 (2011). http://www.sciencedirect.com/science/article/pii/S001046551000531X. Computer Physics Communications Special Edition for Conference on Computational Physics Trondheim, June 23-26, 2010
111. Schlingensiepen, J., Nemtanu, F., Mehmood, R., McCluskey, L.: Autonomic Transport Management Systems—Enabler for Smart Cities, Personalized Medicine, Participation and Industry Grid/Industry 4.0, pp. 3–35. Springer International Publishing, Cham (2016)
112. Schmidt, M.W., Baldridge, K.K., Boatz, J.A., Elbert, S.T., Gordon, M.S., Jensen, J.H., Koseki, S., Matsunaga, N., Nguyen, K.A., Su, S., et al.: General atomic and molecular electronic structure system. J. Computat. Chem. **14**(11), 1347–1363 (1993)
113. Schwarz, K., Blaha, P., Madsen, G.: Electronic structure calculations of solids using the WIEN2K package for material sciences. Comput. Phys. Commun. **147**(1), 71 – 76 (2002). http://www.sciencedirect.com/science/article/pii/S0010465502002060. Proceedings of the Europhysics Conference on Computational Physics Computational Modeling and Simulation of Complex Systems
114. Snavely, A., Gao, X., Lee, C., Carrington, L., Wolter, N., Labarta, J., Gimenez, J., Jones, P.: Performance modeling of HPC applications. In: PARCO, vol. 13, pp. 777–784 (2003)
115. Stanisic, L., Videau, B., Cronsioe, J., Degomme, A., Marangozova-Martin, V., Legrand, A., Méhaut, J.F.: Performance analysis of HPC applications on low-power embedded platforms. In: Proceedings of the Conference on Design, Automation and Test in Europe, March, pp. 475–480. EDA Consortium (2013)
116. Strunk, T., Wolf, M., Brieg, M., Klenin, K., Biewer, A., Tristram, F., Ernst, M., Kleine, P.J., Heilmann, N., Kondov, I., Wenzel, W.: Simona 1.0: An efficient and versatile framework for stochastic simulations of molecular and nanoscale systems. J. Comput. Chem. **33**(32), 2602–2613. https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.23089
117. Subbiah, A., Wasynczuk, O.: Computationally efficient simulation of high-frequency transients in power electronic circuits. IEEE Trans. Power Electron. **31**(9), 6351–6361 (2016)
118. Suma, S., Mehmood, R., Albugami, N., Katib, I., Albeshri, A.: Enabling next generation logistics and planning for smarter societies. Proc. Comput. Sci. **109**, 1122–1127 (2017). http://www.sciencedirect.com/science/article/pii/S1877050917311225. 8th International Conference on Ambient Systems, Networks and Technologies, ANT-2017 and the 7th International Conference on Sustainable Energy Information Technology, SEIT 2017, 16–19 May 2017, Madeira
119. Suma, S., Mehmood, R., Albeshri, A.: Automatic event detection in smart cities using big data analytics. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 111–122. Springer International Publishing, Cham (2018)
120. Taboada, G.L., Touriño, J., Doallo, R.: F-MPJ: scalable java message-passing communications on parallel systems. J. Supercomput. **60**(1), 117–140 (2012)
121. Tikir, M.M., Carrington, L., Strohmaier, E., Snavely, A.: A genetic algorithms approach to modeling the performance of memory-bound computations. In: Proceedings of the 2007 ACM/IEEE conference on Supercomputing, p. 47. ACM, New York (2007)
122. Tomov, S., Nath, R., Ltaief, H., Dongarra, J.: Dense linear algebra solvers for multicore with GPU accelerators. In: 2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and PhD Forum (IPDPSW), April, pp. 1–8 (2010)
123. Tomov, S., Dongarra, J., Baboulin, M.: Towards dense linear algebra for hybrid GPU accelerated manycore systems. Parall. Comput. **36**(5), 232–240 (2010). http://www.sciencedirect.com/science/article/pii/S0167819109001276. Parallel Matrix Algorithms and Applications

124. Usman, S., Mehmood, R., Katib, I.: Big data and hpc convergence: the cutting edge and outlook. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. pp. 11–26. Springer International Publishing, Cham (2018)
125. Vetter, J.S., Alam, S.R., Dunigan, T.H., Fahey, M.R., Roth, P.C., Worley, P.H.: Early evaluation of the Cray XT3. In: 20th International Parallel and Distributed Processing Symposium, 2006 (IPDPS 2006), 10 pp. IEEE, New York (2006)
126. Voorsluys, W., Garg, S.K., Buyya, R.: Provisioning spot market cloud resources to create cost-effective virtual clusters. In: Algorithms and Architectures for Parallel Processing, pp. 395–408. Springer, Berlin (2011)
127. Wolf, F., Wylie, B.J., Abrahám, E., Becker, D., Frings, W., Fürlinger, K., Geimer, M., Hermanns, M.A., Mohr, B., Moore, S., et al.: Usage of the scalasca toolset for scalable performance analysis of large-scale parallel applications. In: Tools for High Performance Computing, pp. 157–167. Springer, New York (2008)
128. Wylie, B.J.N., Geimer, M., Mohr, B., Böhme, D., Szebenyi, Z., Wolf, F.: Large-scale performance analysis of Sweep3D with the scalasca toolset. Parall. Process. Lett. **20**(04), 397–414 (2010). https://doi.org/10.1142/S0129626410000314
129. Yan, S., Zhou, Z., Dinavahi, V.: Large-scale nonlinear device-level power electronic circuit simulation on massively parallel graphics processing architectures. IEEE Trans. Power Electron. **33**(6), 4660–4678 (2018)
130. Yang, R., Gu, L., Tho, C., Sobieszczanski-Sobieski, J.: Multidisciplinary design optimization of a full vehicle with high performance computing. In: Fluid Dynamics and Co-located Conferences, June. American Institute of Aeronautics and Astronautics, Reston (2001). https://doi.org/10.2514/6.2001-1273
131. Zaki, O., Lusk, E., Gropp, W., Swider, D.: Toward scalable performance visualization with Jumpshot. Int. J. High Perform. Comput. Appl. **13**(3), 277–288 (1999)

Chapter 19
Big Data Tools, Technologies, and Applications: A Survey

Yasir Arfat, Sardar Usman, Rashid Mehmood, and Iyad Katib

19.1 Introduction

In the digital world, data are generated on a massive scale and have been used successfully for evolutionary breakthroughs in numerous fields. "Big data" is a progressive term used to define huge volumes of structured, semistructured, and unstructured data sets, which cannot be processed by traditional data management tools and techniques. Big data refers to emerging technologies that are designed to extract value from data, which have four "V" characteristics: volume, variety, velocity, and veracity [1]. One can broadly categorize data sources as external resources (social media, transactional data, emails, etc.) and internal resources (machine-generated data, application logs, etc.). To deal with ever-growing volumes of data, researchers have been involved in developing algorithms to accelerate the extraction of key information from massive volumes of data [2]. Big data technologies are being widely used in many application domains [3–8].

Big data is a wide area of research which co-relates different fields. Figure 19.1 shows an abstract classification of big data.

There are various challenges associated with data transportation where network bandwidth is a bottleneck. Storing huge chunks of data so that one can retrieve it efficiently for data analysis is also a challenging task. Data collected from heterogeneous resources need preprocessing to eliminate redundancy and anomalies and to provide users with a uniform view of heterogeneous data. Keeping redundant data

Y. Arfat · S. Usman (✉) · I. Katib
Department of Computer Science, FCIT, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: yqasim@stu.kau.edu.sa; susman@stu.kau.edu.sa; iakatib@kau.edu.sa

R. Mehmood
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: RMehmood@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_19

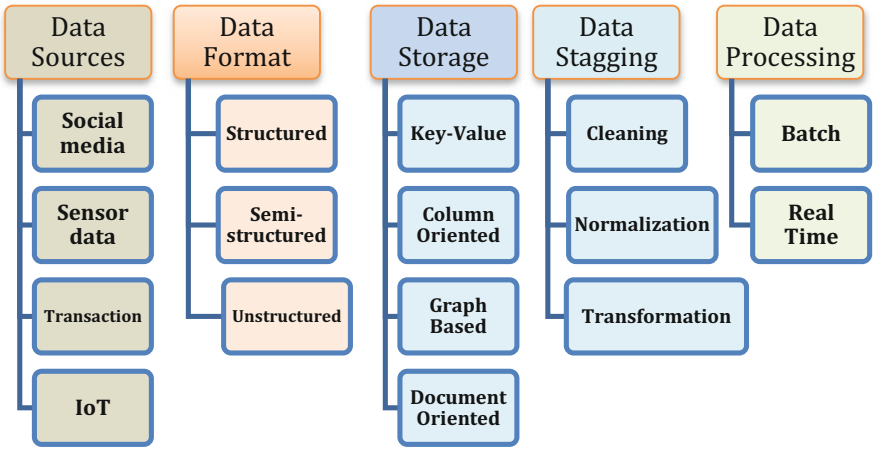


Fig. 19.1 Classification of big data

increases storage resource needs and also affects transmission costs. Unstructured data need to be transformed into a structured format for efficient data analysis. Extracting meaningful information from huge chunks of data specifically in real time is still a challenging task.

The realization of the importance of big data in terms of financial benefits, smart business moves, competitive advantages, and educative decisions has shifted the research focus, over the last decade or so, toward development of big data tools and technologies. With heavy investments pouring in, numerous tools and technologies have been developed to address big data challenges and issues. Most of these solutions are customized to address some specific challenges, and it is very difficult to choose which option among the plethora of tools and techniques is most suitable for any particular situation. Some research efforts have been dedicated to providing insights into these tools and techniques. For example, Yadranjiaghdam et al. [9] have focused on tools and techniques most suitable for real-time or near-real-time analytics of big data. Oussous et al. [10] have provided an abstract view of different big data techniques and drawn comparisons on the basis of different system layers. Chen et al. [11] have reviewed opportunities and challenges of big data in general and have also addressed various tools and techniques to cope with these challenges.

Numerous tools and technologies have emerged over the years to deal with big data challenges, e.g., MapReduce, Spark (in-memory computation), and MongoDB. This chapter presents a summary of state-of-the-art tools and methodologies for the processing of big data applications. This overview consists of a discussion about the objectives, methodologies, and key approaches in existing techniques and their critical analysis. We also highlight some of the key applications of big data and provide a detailed review and taxonomy of research efforts within each application domain.

The rest of this chapter is organized as follows. Section 19.2 provides an overview and taxonomy of numerous open-source tools and technologies, developed over the years to deal with challenges associated with big data. Section 19.3 provides a taxonomy and review of commercial tools and technologies for big data. Section 19.4 highlights some of the key applications of big data: healthcare and smart city applications, genome sequence annotations, and graph-based applications. We also provide a taxonomy of research efforts within each application domain. Section 19.5 concludes the chapter.

19.2 Open-Source Big Data Tools and Technologies

There are numerous open-source tools and technologies available in the market for the processing of big data. Each tool or framework has its own characteristics, limitations, and features.

19.2.1 Hadoop

Hadoop [12] is open-source software for processing big data and has certain characteristics: scalability, reliability, fault tolerance, high availability, local processing and storage, distributed and parallel computing, and cost effectiveness. It uses a simple programming model for the processing of large data sets across computer clusters. It was developed to scale up from a single server to thousands of machines and provides colocation of storage and computation. Its library was designed in such a way that it can self-identify failures and provide a recovery mechanism. The Hadoop file system (HDFS) [12] is an important component of Hadoop and has master–slave architecture. It consists of two components: NameNode and DataNode. The distributed file system delivers high throughput. It can process large data sets and was designed for a low-cost system. The main objective of HDFS is to provide the following features: increased throughput by decreasing network congestion, fault detection and isolation, access to large data sets, access to streaming data, a simple coherency model, and portability between different hardware and software.

19.2.2 MapReduce

MapReduce [13] an open-source framework for the processing of a huge amount of data. MapReduce consists of two basic components: the map phase and the reduce phase. The workload is divided into smaller workloads in the map phase, and the output, in the form of a key–value pair list, is passed (shuffled) into the reduce

phase, which analyzes and reduces the results to produce the final output. It splits the data set into pieces, which will be processed in a parallel manner by the map task. Mapper output is sorted by the framework and further sent to the reduce task. Both output and input are stored in separate files. This framework also examines task monitoring and scheduling that have failed during execution, and re-executes them. MapReduce has been widely adopted because of its simplicity, scalability, speed, recovery, and minimal data movement. It has various limitations: creation of bottlenecks due to reading of the disk again and again, inefficiency for iterative processes, and the fact that it is not primarily designed for iterative processes.

19.2.3 YARN

YARN [14, 15] is an open-source framework for job scheduling and cluster resource management. It was initially designated by Apache as a reformatted resource manager. For big data applications, YARN is nowadays categorized as a large-scale distributed operating system. The essential theme of YARN is to divide the functionalities of resource management and job scheduling and monitoring into distinct daemons. The data computation framework is formed by the node manager and resource manager. The fundamental authority that determines resources between all applications in the system is a resource manager. The main responsibility of the node manager is to monitor the usage of resources, such as the network, memory, central processing unit (CPU), disk, etc. It has features such as multitenancy, compatibility, scalability, and cluster utilization.

19.2.4 Apache Pig

Apache Pig [16] is an open-source platform for analyzing a large data set that has high-level language (HLL) representation. The main characteristic of Apache Pig is a structure that is flexible for substantial parallelization, which allows handling of large data sets. Currently, the infrastructure layer of the Pig has a compiler and a production sequence of map and reduce programs. The language layer of the Pig consists of a textual language, called Pig Latin, which provides ease of programming, optimization opportunities, and extensibility.

19.2.5 Apache Hive

Apache Hive [17] is open-source software that enables management of large data sets and querying of data in a distributed storage environment. It provides facilities such as ad hoc queries, analysis of enormous data sets, and summarization of the

data store in Hadoop. It has a data warehouse infrastructure that builds on the Hadoop and query data using an SQL-like language called HiveQL. When there is difficulty in expressing the logic in HiveQL, it also enables the MapReduce developer to customize the map and reduce phases. It provides a quick response to extensive data sets, is scalable, and is very extensible. Data extraction, transformation, and loading (ETL) are easy, and queries are executed using MapReduce.

19.2.6 Apache Storm

Apache Storm [18] is an open-source distributed real-time computation system for the processing of big data. It is straightforward; any programming environment can use it, and it is beneficial to YARN for machine learning, ETL, continuous monitoring of operations, distributed remote procedure call (RPC), and real-time analytics. Its scalability, fault tolerance, reliability, speed, and ease of operation make Storm ideal for real-time processing of data.

19.2.7 Spark

Spark [19, 20] AB is an open-source tool for processing large data sets. Spark was introduced to overcome the disk I/O and performance issues of Hadoop. It has several features (such as in-memory computation) that make it unique, and it provides an in-memory data-caching facility. Spark supports several programming languages (Python, Java, and Scala) for processing large volumes of data. It performs processing on massive data sets very quickly. It processes data 100 times faster than Hadoop and MapReduce. It has an execution engine—a direct acyclic graph (DAG)—which manages acyclic data flow, and in-memory computing. It also has the characteristic of speed to perform tasks very fast, ease of use, support for several platforms, and generality, which means, for example, that SQL, GraphX, and DataFrame can be combined in a single application. It also runs anywhere, meaning we can access data from HBase, Hive, and Tachyon. One limitation of Spark is that memory consumption issues are not easy to handle in a user-friendly way. It takes more memory than other big data platforms.

19.2.8 Spark Streaming

Spark Streaming [21] is a component of Spark [22]. It is a core application programming interface (API) of Spark, which makes streaming processing of data more fault tolerant and scalable, and provides high throughput. It uses complex algorithms to process data sources such as Twitter, Kafka, Kinesis, and TCP sockets

for advanced methods such as Joins, MapReduce, etc. The processed data can also be presented in the dashboard, database, and file system. Moreover, we can also apply graph processing and a machine learning algorithm of Spark on the data stream. It also provides fault tolerance, Spark integration, ease of use, support for different programming languages (such as Java, Python, and Scala), and deployment options, and it can read data from ZeroMQ, Twitter, HDFS, etc. In short, Spark streaming enables the streaming application to be scalable, reliable, and fault tolerant.

19.2.9 Blink DB

Blink DB [23] is used for processing of large-scale data for interactive SQL queries. It enable clients to adjust the accuracy of a query for a quick response time and allows collaborative queries of large data by executing the queries on the sample data. There are two main features of Blink DB: first, it has an adaptive optimization framework, which constructs and keeps multidimensional data samples from the original data over time; second, it has a strategy for appropriate sample selection, which is dynamic, on the basis of the query's response time and accuracy needs. The main goal of Blink DB is to support interactive queries, i.e., aggregate queries on large volumes of data.

19.2.10 Berkeley Data Analytics Stack

Berkeley Data Analytics Stack (BDAS) [24] is an open-source software tool for processing big data. It was developed by AMPLab, combining different components of software for application of big data. It has five layers: resource virtualization at the bottom, storage, a processing engine, access, and interfaces. On each layer there are different components. For resource virtualization it has two components (storage, open source): Mesos [25] and Hadoop YARN [26]. Mesos also provides features such as scalability, fault tolerance, and scheduling of multiple resources such as ports, memory, disk, and the CPU. The next layer is the storage layer, which consists of Tachyon, HDFS, Succinct, S3, and Ceph. Tachyon [27] is a storage system, which is memory centric, allowing the user to share data reliably on a very high-speed cluster framework. It has higher performance than HDFS and has a caching mechanism, which enables it to decrease memory access. On the processing layer it contains the Spark core, which has great features such as speed, ease of use, and generality. The components are BlikDB, SharkSQL, GraphX, MLBase, MLib, SparkR, Sample Clean, Splash, MLpipelines, Velox, and Spark streaming. On this top layer there are many wrappers of Spark such as Spark streaming (a large-scale, real-time processing system), MLBase (a distributed machine learning library for Spark) [28], MLib (a machine learning library) [29], SparkSQL (a module of Spark,

which works on structured data) [30], SparkR (which contains a distributed data frame implementation), GraphX (a resilient distributed graph system on Spark) [31, 32], and BlikDB (queries with bounded errors and bounded response times for large volumes of data).

19.2.11 MongoDB

MongoDB [33] is an open-source tool for processing large volumes of data stored in a database. It has key features such as automatic scaling, high performance, and availability. In automatic scaling it has horizontal scaling, which is the main fragment of core functionality. The data are distributed among clusters using automatic sharing. The set of replicas delivers low latency and high throughput. It has high availability as it contains replica sets that provide recovery upon failure automatically. For high performance, it has indices, which make the querying process very fast. Moreover, it minimizes I/O activity in the database by embedding different models.

19.2.12 R

R [34] is an analytical tool for big data applications. It is used for statistical analysis and graphical visualization. R offers a broad range of statistical analysis: classification, classical data testing, clustering, analysis of time series, graphical methods, nonlinear modeling, and linear modeling. It handles data in an effective manner and provides robust data storage. It guarantees a smooth procedure for matrices and vector data so that statistical calculations can be optimized. It has the R package, which is a comprehensive R archive network (CRAN), so that the user can get any required statistics. In the statistical field it can be promoted as open source. It is very supportive of many platforms such as Mac OS, UNIX, and Windows.

19.2.13 Dryad

Dryad [35] enables developers and programmers to utilize computer resources efficiently. It also allows users to use many machines with multiple cores. It is a framework for parallel processing of data, establishing a path among the programs in the form of a graph. Often, a programmer writes the function of a MapReduce and use Dryad to create a graph that evaluates the conforming data. Dryad processes the data in a DAG manner. This framework has sufficient function for processing big data instead of using parallel data operations. There are some obstacles to using the

system; inexperienced programmers and the efficiency of data mining and analysis can be issues. Consequently, there is need to develop new techniques for efficient processing of data.

19.2.14 High-Performance Computing Cluster

A high-performance computing cluster (HPCC) [36] is an open-source platform for the processing of huge volumes of data sets. It enables the user to process big data efficiently and effectively. It is more reliable than other platforms. This platform provides scalability, high performance, and agility. It has a real-time data delivery engine for data warehousing and query processing. It has a very powerful programming language for the processing of large data sets. For big data queries it has a platform based on standard web services. HPCC provides various features that are necessary to overcome the challenges of big data. It can run commodity hardware and also has a distributed build in the file system, fault tolerance, an integrated development environment (IDE) for development, different modules such as machine learning, and operation tools. It has three main components [37]: the HPCC data refinery is an ETL engine that enables the user to integrate and manipulate data; the second component is a data delivery engine, which provides low latency, fast query response, and high throughput; and the third component is an enterprise control language (ECL), which distributes the workload among nodes, with a library for machine learning development and synchronization of algorithms. HPCC has Thor and Roxie; on the other hand, Hadoop has MapReduce and HDFS for processing.

19.2.15 Neo4j

Neo4j [38] is a graph-based database, which is an open-source database for processing large volumes of data. It has its own query language and has two important graph technologies: graph-based storage and a graph-based processing engine. The graph-based processing engine provides native graph processing. Since the nodes are physically connected to the graph, it provides an efficient means of graph processing. On the other hand, it provides native processing and storage for graph-based databases. It provides better scalability than other databases. It has the following characteristics: vertical scaling, higher performance, and concurrency. It also supports various platforms such as Java, Python, Ruby, Net, and PHP, which make it more useful for developers. It also has some limitations; for example, there is no support for sharing. Moreover, there are also limitations on node properties and relationships.

19.2.16 Pentaho

Pentaho [39] is a data integration and business analytics tool. It is an open-source platform and is very popular for blending, analyzing, and visualizing big data. It has wide capabilities for data mining and analysis. It is an alternative choice for developers and offers a well-updated user interface. It provides native support for Hadoop and any type of data source. Pentaho users do not need to write code for integration. It also provides several other features for users, e.g., business intelligence (BI), data integration, dashboarding, ETL capabilities, online analytical processing (OLAP) services, and data mining. The limitations of Pentaho include limited data visualization, lack of proper documentation, and limited analytical tools, which require more improvement.

19.2.17 Talend

Talend [40] is an open-source platform for processing big data. The architecture of Talend covers all user needs regarding integration and governance of data. It has various features such as scalability, ease of use, and reliability; these features makes it very suitable for developers. The tools in Talend consist of products for development, data management, deployment, and integration of products. There are many advantages of using Talend; with its ETL tool it can equalize burdens across the server processing on the cluster. It also uses Jaspersoft BI software. The interface of ETL provides support for importing metadata. Its configuration and linking of various components enable developers to generate more productivity than any existing programming language. However, it also has limitations such as the need for Java Database Connectivity (JDBC) for source access. There is also no product for metadata management and quality of data, and there are bottlenecks in job automation, data partitioning and repartitioning, and allocation of resources across the grid.

19.2.18 Oracle

Oracle [41] provides a wide range of tools for big data. It has its own platform for big data: Oracle Exadata. It uses the R platform for advanced analytics, in-memory computation, and data warehousing. Oracle provides numerous software products, e.g., the Oracle NoSQL Database, Oracle Data Integrator and Loader for Hadoop, and Oracle R Enterprise tool.

19.3 Commercial Tools and Techniques for Big Data

19.3.1 Syncfusion

The Syncfusion [42] platform for big data is designed for Windows. There are several challenges associated with big data processing and management. This platform provides very useful features, including query processing for structured and unstructured data, and it provides cost-effective storage. With linear scalability we can store any type of data on the commodity hardware. Syncfusion has made these dominant technologies available on Windows. Using this platform we can access the Hadoop framework completely. There are many companies using this framework: Adobe, Yahoo, Facebook, Hulu, Microsoft, and Amazon.

19.3.2 Cloudera

Cloudera [43] is a commercial platform for processing big data. It provides Impala, which deals with real-time, massive big data processing for Hadoop. It provides a very secure, simple, and fast platform for Hadoop. It helps the user to solve the most challenging issues related to the business domain. Moreover, Cloudera also contains the core components of Hadoop: MapReduce, YARN, and HDFS. It has several features, such as flexibility, integration, security, scalability, high availability, and compatibility [44]. It has distributed computing, a web-based interface, important enterprise capabilities, and scalable storage.

19.3.3 Pivotal HD

Pivotal HD [45] is an important component of Apache Hadoop for analysis of big data. It uses Hadoop's native tools for processing big data. It supports more flexible models such as various packages for commodity hardware. Because of this flexibility, it addresses various enterprise concerns over security, performance, regulatory aspects, control, cost, etc. Pivotal HD offers backup and recovery, data protection, georedundancy, and robust availability in case of failure.

19.3.4 EC2

EC2 [46] is an interface for web service that provides the user with easy configuration and obtains results with minimal friction. It allows the user to run the computing environment of Amazon and to have complete control over the computing resources.

It minimizes the required time for accessing the server and is compatible in terms of computing requirements. It allow the user to pay only for the capacity that they occupy. It provides fault tolerance and recovery in case of failure. It provides several features, such as flexibility, security, scalability, and reliability. On the other hand, it is very costly, and many users cannot afford it.

19.3.5 Tableau

The Tableau platform [47] provides analysis, reporting, and visualization of data. The main goal of the Tableau framework is to provide fast and interactive reports. It is a very popular and well-adopted BI tool among nontechnical and technical users, for various reasons. Firstly, it provides self-service BI in which users of BI and business analysis can create their reports. It also enables quick development and has drag-and-drop features allowing any type of user to build their own dashboard. Lastly, it has data visualization, which permits the user to perform effective analysis and visualize trends. It supports various platforms such as Python, XML, API, and JavaScript. Its features includes an excellent user interface, integration, mobile support, customer services, a user forum, low cast, and ease of upgrading. On the other hand, it has limitations in preparation of initial data, and statistical features are avoided in this framework, including financial reporting.

19.3.6 IBM Info Sphere

IBM [48] is a vendor of big data. It provides various platforms for integration and building of big data warehouses and has various products for users. It also has BI, with capabilities for big data such as computing of streams, solutions of data warehousing, and Enterprise Class Hadoop. The InfoSphere stream is integrated with statistical packages and social sciences, including capabilities for dynamically changing real-time data.

19.3.7 SAS

SAS [49] provides various techniques for analysis of big data, providing an infrastructure for high-performance analytics and statistical software. It provides features such as distributed processing, grid commuting, database analytics, and in-memory computation. It can do deployment in the cloud and on-site. It also provides solutions to complex problems and is an advanced analytical tool used in leading industries.

19.3.8 Teradata Aster

Teradata Aster [50] is a multidisciplinary advanced analytics platform with powerful analytical solutions that provide useful data insights at scale and provide a variety of advanced analytics techniques, e.g., data visualization and R packages.

19.4 Key Applications of Big Data

19.4.1 Healthcare and Smart City Applications of Big Data

Many governments are implementing smart city ideas to improve the living standards of their people. The smart city uses various technologies to enhance the performance of transportation, health, water services, education, and energy, leading to higher levels of relief. Big data analytics is the latest technology because of the large volumes of data that are digitized, as they are relevant to healthcare, education, etc.

Nuaimi et al. [51] surveyed the various applications of big data for a smart city. They compared the different definitions of big data for the smart city by highlighting some of the core challenges and issues in big data, e.g., security and privacy, the smart city population, cost, data quality, data and information sharing, and data sources and characteristics. The "smart city" and "big data" concepts can be integrated with each other for better reliance, quality of life, and efficiency of management systems. Various authors have coined different requirements for big data for smart cities and their applications, i.e., management of big data, a platform for processing of big data, an infrastructure for the smart network, advanced algorithms, and security and privacy.

The primary goals of healthcare are diagnosis and treatment of ailments, and disease avoidance, through proper medication. Presently, healthcare is reaching a higher level on which we can apply IT to health informatics. In the past, there were lots of issues involved in managing large volumes of data; nowadays, big data tools offer health informatics. The arrival of health informatics has been a significant change in the area of healthcare. Traditionally, relational database management systems (RDBMS) were used to find hidden values, but they have several limitations related to lack of processing of unstructured data, fault tolerance, and linear scalability. Hadoop gives better results for large volumes of data than RDBMS. In many situations, RDBMS solutions fail as the volume increases.

Information and communications technology (ICT) has three primary components: cloud computing, the internet of things (IoT), and big data. If we combine

the features of these elements, it is possible to reshape the next generation of the e-health system. Suciu et al. [52] attempted to analyze the current components and techniques, securely integrating big data processing with a cloud machine-to-machine system based on remote telemetry. They also described various issues in state-of-the-art methods for developing health applications. To overcome these problems, they proposed a machine-to-machine system-based decentralized cloud architecture, with a general system and remote telemetry units (RTUs), for e-health applications. They developed this system for big data and collected information from sensors with huge volumes. They found that multiple sources can be implemented seamlessly using real-time data from cross-domain applications and using the inferred content scalability, and energy efficiency can be achieved. In the coming years, big data will have a huge effect on the field of health. It is possible that more data will be gathered and merged from different information sources with automated evaluation. Moreover, possession of biomonitoring and lifestyle data will enable health interventions to be tailored more to individuals. Suciu et al. [52] also argue that big data has the potential to advance health economics as a discipline.

Collins [53] analyzed SWOT (strength, weakness, opportunities, and threads) by using big data techniques, and noted that big data have more opportunities and strengths for healthcare. Consumer behavior can be monitored from larger data sets through accurate analysis. Collins [53] also described different strengths of health economics that, with an availability of more data, could easily identify which medicine is useful for particular individuals. Hence, this will increase reliable decision making for health improvement. People from different backgrounds can use open data for analysis. The weaknesses are that it is very costly to store the data and to manipulate huge volumes of data. With big data, there are a lot of opportunities in having a larger data set for communication with everyone and for generation of better and more sophisticated analysis. One of the major threads is privacy of data in that people feel that their data may be misused. On the other hand, big data offers a lot of benefits for the individual, such as better health monitoring, smart health solutions, and fewer mistakes. It has been concluded that there is a need for analytical skills in big data. There should be more chances to make the health system more useful in tailoring medicine and providing more knowledge for individuals.

Raghupathi et al. [54] presented a comprehensive overview of big data analysis for healthcare specialists and scientists. They described the various possibilities and capabilities of big data for healthcare analytics. They stated that big data in healthcare refers to an electronic data set that is complex and hard to manage even with state-of-the-art software and hardware. Moreover, they also stated that the benefits of big data in healthcare are early detection of disease symptoms, ability to apply more effective treatments, and ability to detect healthcare fraud more quickly. Big data can be helpful in individual sectors of healthcare, such as clinical operations, research and development, public health, evidence-based medicine, genomic analytics, and patient profile analytics. Raghupathi and Raghupathi [54] proposed an architectural framework for healthcare, consisting of four components:

big data sources, big data transformation, a big data platform and tools, and big data analytics. The big data sources are both internal and external, with multiple locations, applications, and formats. Big data transformation consists of ETL; in this phase, data are transformed into unique formats. The big data platform and tools are Hadoop, MapReduce, HBase, and Hive, and these can be used to process the data. The big data analytics consists of queries, reports, OLAP, and data mining. Raghupathi and Raghupathi [54] also presented different challenges that a big data analytics platform in healthcare must resolve for processing the given data. There should be criteria for evaluation of the platform; it must have ease of use, availability, security and privacy, continuity, and quality assurance. They concluded that applications of big data in healthcare are at an early stage of development, but they believe that more advancement in big data tools and platforms can accelerate their development process.

Mehmood and Graham [8] presented a model for healthcare transport capacity sharing. Nowadays, the healthcare industry is concentrated on investigating reasons for failure in the primary quality control process, customer needs, services, and duplication logistics. Mehmood and Graham [8] explained the logistics as analysis and modeling of transport and distribution systems through large data sets created by Global Positioning System (GPS), combined with human-produced activity. They also noted that logistics firms need more technical support regarding the three V's of big data. The main objective of this research was to provide more awareness about capability sharing and optimization from a smart city perspective. There is a need to develop new tools and methods to overcome the challenges of big data in the smart city. The main goal was to contribute to improvements in transport capacity sharing through big data. Mehmood and Graham [8] explained that there are many problems in transportation, as there is poor coordination of transport, lack of vehicles, and poor maintenance and repair. To overcome these issues, there is an enormous need for a new approach to provision of carriage. The authors have therefore introduced a new framework, combining concepts from the literature on the smart city, big data logistics, and capacity sharing. They have also developed a Markov model by matching the needs for transport of patients with healthcare transport service provision. The fundamental purpose of this model is sharing of transport capacity in a smart city to enhance efficiencies in meeting patient needs for the city healthcare service. For analysis, the authors considered 13 different scenarios and found that the likelihoods of system failure and performance variance tended to be highest in a scenario of highest demand with zero sharing.

Huang et al. [55] reviewed different applications of big data in health sciences, including various systems with sources of big data; these were recommendation systems, epidemic surveillance based on the internet, food-based monitoring, sensor-based health condition monitoring, inferring air quality using big data, genome-wide association studies (GWAS), and expression quantitative trait loci (eQTL). The main purpose of these systems is to collect data from various people and make the analysis much faster than is possible through official channels. Huang

et al. [55] also suggested that to start any big data project we have to follow some steps; the first one is to choose the right problem. The second step is to find the sources of the data, which can be collected via the internet, smart devices, hospitals, social media, and omics profiling. In the third step, data collected in the previous step are stored in NoSQL, GEO, and dbGaP. The next step is to create a report of the analysis using vivid visualization. Data can also be analyzed using three different systems—collaborative filtering, content-based filtering, and hybrid filtering—and through deep learning and network analysis. For visualization of big data results, there are various tools that can be used: R, Circus, Gephi, Tableau, etc. Each of these has its pros and cons. Huang et al. [55] also described that before the era of big data, people obtained information through television, newspapers, and the internet, and it took years to create awareness about healthcare. However, in the big data age, information is pushed directly to people via smart devices.

Barkhordari et al. [56] noted that nowadays there are huge volumes of data being generated. Traditional management systems can support analysis in various fields, including social networks, medical networks, scientific instruments, and meteorology. How we retrieve, store, and capture data or information is a fundamental problem in traditional systems. To overcome these issues, there is need for scalable and distributed solutions to these kinds of problems. To overcome all of these problems, Barkhordari et al. [56] proposed a ScaDiPaSi—a scalable, distributable technique for exploring patient similarity. In this method, they used various data sources; unlike other researchers, they did not concentrate on structured and semistructured data sources. For two patients, the same lines of evidence may involve different items. All of these formats can be retrieved by data integration. It is a dynamic method to store information about patients and efficiently distribute it on the hardware node. For analysis of the proposed technique, the authors did an evaluation based on the execution time and accuracy of the ScaDiPaSi method. By using the ScaDiPaSi method, they found that they could quickly achieve the desired results using a distributed and scalable structure. They also found accuracy of up to 63% in their proposed scheme.

Toga et al. [57] presented a framework establishing reasonable and practical data-sharing policies that incorporated the sociological, financial, technical, and scientific needs of a maintainable big data community. They also argued that there are enormous challenges involved in sharing big data. Key features of big data are the volume of the data, incompleteness of data, incompatibility of data, incongruence of sampling, and heterogeneity of data. Sharing of big data requires innovative policies and clear guidelines that can enhance cooperation instead of causing other problems and complexities. To overcome the complexities, Toga et al. [57] introduced a big data policy framework, which consists of various suggestions for sharing big data depending upon the domain of application. These proposals are (1) policies for storing and securing data and ensuring human subject protections; (2) processes and policies for data sharing; (3) protection of data from unauthorized access; (4) agreements for data usage; (5) data value, as sharing data

that are incomplete will have less value; and (6) big data–sharing policies for achieving cost efficiencies. Many healthcare and biomedical studies have large, incongruent, and heterogeneous data sets. Thus, there are many important barriers—technical, social, and regulatory—that need to be overcome to ensure the power of big data. The authors concluded that there is a need for implementation of the aforementioned policies for sharing of data and security of personal information, which will have long-term impacts on big data analytics. In addition, there have been numerous research studies focusing on smart infrastructure [2, 58], healthcare [59–62], transport [6, 63–69], and other applications [70, 71].

19.4.2 Graph-Based Applications of Big Data

Jun et al. [72] noted that in data-intensive applications, there are great needs for high-performance computation and massive resources. In many scientific and engineering applications, people and researchers are interested in a subset of the whole data set so they can access it frequently. These groups of data can be processed by specific domain applications. As an assumption, if two pieces of data have been processed at the same time, it is a strong possibility that these data will be processed as a group in the future. For simplicity, Hadoop uses a random placement method for load balance. In MapReduce and Hadoop, default random placement does not count the semantics of data grouping. A cloud cluster is grouped into many small groups, limiting the degree of parallelism for the data and resulting in a performance bottleneck. Jun et al. [72] also argued that other methods proposed for data locality are not effective, because of cost and overhead issues. Their observation was that random data placement is affected by three factors. The first is that each replica should have a data block on each rack, the second is that there should be a maximum number of map tasks on every node, and the last is data grouping access patterns. The authors presented a new "data group aware data placement scheme" (DRAW). It takes runtime data group patterns and equally distributes the data. It consists of three phases: firstly, there is data group information learning from the log. A history data access graph (HDAG) is used to approach the files; it is based on the history. In the Hadoop cluster rack, the NameNode maintains a log of the history of each operation, including the accessed file. However, this scheme also has certain issues. The first issue is that the log is huge, involving a traversal latency problem; the second issue is that frequently accessed files are not similar. Therefore, to resolve these issues, there is a need for a checkpoint to traverse the log of the NameNode. Secondly, there is clustering of the data group matrix (DGM), which shows the relationship between data blocks. It can be generated on the basis of the HDAG. Thirdly, data group reorganization is based on the optimal data placement algorithm (ODPA). The ODPA is based on the submatrix for the cluster data–grouping matrix. The authors also proved that Hadoop random placement of data is not efficient. They

did an experiment with real-world applications; it reduced the completion time of the map phase and the execution time of MapReduce.

Lee et al. [73] noted that a graph is essential for discovering knowledge from a given data set. They stated that there are two dimensions of graph analysis. One is graph mining (GM); it has a central focus on automatic knowledge discovery. The second is online graph analytical processing (OLGAP); its main concentration is pattern matching on a subgraph. Both dimensions are complementary and concentrate on solving complex problems. It is difficult to process multiple graphs and send the results to the system. The authors proposed enabling graph-mining abilities in the Resource Description Framework (RDF) triple store. For achieving the desired goal, they implemented six different graph-mining algorithms using SPARQL. For evaluation of the proposed technique, they used various computing environments and nine different data sets. Their assessment showed scalable performance for real-world graph analysis.

Xia et al. [74] stated that big data analytics are necessary for entities that can be easily represented in the form of a graph. For analytics of a large-scale graph, there is the main problem of delivery of effective solutions with irregular data access. It is the main challenge for the processing of computation of graph-based patterns. The major challenges Malewicz et al. [75] that big data faces for graph processing are performance, a large volume of data, and irregular data access. Big data tools are not suitable for processing of a graph, for reasons such as scalability, architecture awareness, and systematic optimization. To overcome these issues, Xia et al. [74] proposed a system called G. It enables the user to organize the data for the architecture of parallel computing. It also consists of visualization, graph storage, and analytics. The authors also analyzed the data locality regarding graph processing and its effects on the performance of the cache memory on a processor. They also measured the traversal performance of the graph by both serial and parallel execution. For the experiment, they analyzed the parallel system and the G system that was proposed. They showed that the G system performed much better than traditional systems.

Fang et al. [76] proposed a new technique called a bipartite request dependency graph (BRDG) for investigation of the relationship between objects on the web. They also leveraged the MapReduce programming model for construction of the BRDG from the large network data. The nodes of the BRDG have two types of objects: primary objects such as the URL in a web browser, and secondary objects such as those that trigger the primary objects. It is challenging to derive the structural features of the BRDG. To overcome this difficulty, Fang et al. [76] proposed a coclustering algorithm; from the BRDG it extracts the coclustering coherence. In coclustering of BRDG, each vertex signifies the strong connection to the bipartite subgraph. A parallel tri-nonnegative matrix factor (tNMF) algorithm was designed and implemented. The primary purpose of this algorithm was to provide efficient large-scale graphics decomposition. For the experiment, the authors divided the subgraph into four structural patterns: click star, embed star, single layer mesh, and

multilayer mesh. They noted that the multilayer mesh is a very famous structural pattern.

Xue et al. [77] proposed a new technique based on bipartite graph–oriented locality scheduling (BLOS) for MapReduce frameworks. For improving the locality of the scheduling approach effectively, it is necessary to have the tasks and related data on the same node. Xue et al. [77] also argue that data locality refers to a task that is obtained from the same local machine. A higher degree of local localization can enhance the performance of the system. Because of the localization, it minimizes the execution time of the job and reduces the communication time. Moreover, they described that there are three different types of priority according to proximity principles. The first priority will be given to the local task within the node, the second priority will be given to the task within the rack, and the third priority will be given to off-rack tasks. However, there is a problem regarding the locality optimization; this approach creates another problem of global optimization instead of local optimization. Xue et al. [77] found that recent studies were not sufficient to tackle the data locality issues. To overcome these issues, they proposed a new scheduling algorithm called Bipartite-Graph Oriented Locality-Aware Scheduling (BOLAS) for MapReduce tasks. BOLAS can operate in any environment whether it is homogeneous or heterogeneous. The main aim of BOLAS is to enhance the data locality without affecting the efficiency of execution. BOLAS starts solving the problem of scheduling of data locality by matching the bipartite graph in it and trying to allocate the data close to the task. BOLAS uses a global data placement method; through this, better performance can be achieved without affecting the nodes. It also avoids network congestions without generating local nodes. The design of the algorithm consists of two parts: one is resource modeling and the second is computing node performance estimation. In resource modeling, MapReduce has two essential resources: data blocks and NameNodes. In performance estimation of the computing node, BOLAS dynamically dispatches the blocks according to the performance of the particular node. BOLAS also builds a bipartite graph in various cases by adding the virtual blocks and nodes if they are needed. Then the KM algorithm is applied for an optimal matching result with minimum weight. The benefit of BOLAS is that it has high data locality. It also increases the throughput of the cluster system and minimizes the congestion of the network. In their experiment, Xue et al. [77] evaluated the ratio of data locality, execution time, and network bandwidth. The results showed that the data locality was improved with the proposed technique by nearly 100% and the execution time was reduced by up to 67%.

Orozco et al. [78] described various challenges in stencil application for computations. The main limitation of these applications is off-chip memory access in terms of the latency and the data. To overcome these problems, they proposed locality optimization based on data dependency graphs for stencil applications. For this purpose, first they presented a formal description of data that were not generated from the source code. They proved this by using a mathematical method

for the computation power and the bandwidth of the multicore architecture. For the experiment, they used various approaches such as Naïve, overlapped, split, and diamond tiling. They found that the diamond tiling technique was better than the other approaches.

Nazarabadi et al. [79] presented a locality-aware skip graph to overcome the issues of the name ID assignment method of the skip graph's node. Skip graph locality assigns name IDs to nodes such that closer the two nodes are to each other, the more similar the prefix in their name IDs will be. Nazarabadi et al. [79] also argued that state-of-the-art methods do not consider the skip graph's node locations. The main objective of the proposed technique was to minimize the latency in the search query from end to end. The authors proposed a dynamic and fully decentralized algorithm, DPAD. The results showed an 82% improvement in the data locality and a 40% improvement in search query end-to-end latency.

Kandemir et al. [80] provided a solution for optimal memory layout detection. The proposed approach has two basic elements: one is the construction of the problem in a special graph structure, and the second is the use of a Integer Linear Programming (ILP) solver to define the memory layout. It is the first approach that allows dynamic changing of the layout cache locality. The authors also showed that the proposed framework for the locality is persuasive, but there is a need for dynamic layout modification for large applications.

Chernov et al. [81] described the problem of thread partitioning of sequential programs. They proposed a new algorithm to overcome this issue. They also argued that performance could be improved by considering the locality of the program. Many programs have locality characteristics by nature, so they combined two optimizations for the new algorithm. The first is that the nonloop region can be parallelized. The second is that the partitioning is performed in such way that data locality can be improved in a new thread. For the evaluation of their approach, they generated a data dependency graph (DDG) and showed that their proposed technique is feasible.

Zhang et al. [82] noted that k-nearest neighbor (k-NN) is a simplified approach for various applications, especially machine learning and graph-based applications. Besides its multiple characteristics, it has computational complexity. To solve this problem, they proposed a new algorithm that divides the whole data set into small groups, and it makes sure that each item in k-NN belongs to the group. This leads to accuracy and fast speed. To make the proposed algorithm more accurate, they divided the groups in such a way that similar items remain in the same group. Secondly, they kept the group size small as possible. For the groups, they also proposed locality-sensitive hashing (LSH), which guarantees rigorous performance even for the worst-case performance. The proposed approach can efficiently generate a graph with good accuracy.

Zhang et al. [83] noted that a widely explored area of graph databases is similarity in graph processing. They also argued that graphs have an important role in various applications such as pattern recognition and information retrieval. They identified

two categories of the graph search: one is a subgraph search and the second is a similarity search. In their paper, they explored the k-NN similarity search problem, using locality sensitivity hashing. They proposed an algorithm for a fast graph search that transforms complex graphs into vector representations based on the prototype in the database. After this, it also accelerates the query efficiency in the Euclidean space by employing LSH. The authors evaluated their approach against real data sets, achieving high performance in terms of accuracy and efficiency.

Yuan et al. [84] pointed out that there are two main challenges in large-scale graph processing: one is lack of efficient storage and the second is lack of locality access. They also noted that there are various popular approaches for processing of graphs and storing the data, such as the storage-centric, vertex-centric, and edge-centric approaches. They listed common graph partitioning approaches such as vertex cuts and edge cuts. A commonly used partitioning scheme for processing of a graph is vertex-centric hashing. However, these approaches have issues of poor locality and communication overheads. To solve these issues, Yuan et al. [84] proposed a new path-centric approach for fast iterative graph computation of extremely large graphs. It is a path-centric approach for both the storage and processing tiers. It is an efficient approach for storage and structure design on the storage tier. It also allows the fast inner edge and outer edge loading for gathering and scattering the parallel computation of the graph. For the computation tier, its processing is used in such a way that optimizes the locality of the large-scale graph. It iterates the processing at the chunk level and partition-level computation. A work-stealing approach was introduced to balance the workload among parallel threads. For evaluation of their approach, the authors chose a real data set in which they were able to demonstrate the performance of the proposed approach in terms of better balance and speed-up.

Shao et al. [85] proposed a new approach of a partitioning aware graph computation engine (PAGE). The benefits of this method are that it controls the online graph partitioning statistics of the underlying results of a graph, it monitors the parallel processing resources and enhances the computation resources, and it is designed to support various graph partitioning qualities. In evaluation of the chosen scheme, the authors showed that it performed well under various partitioning approaches with different qualities.

Qin et al. [86] noted that mining of dense subgraphs from large graphs is a fundamental mining task that can be applied to various application domains such as network science, biology, graph databases, and web mining. They also argued that existing schemes concentrate just on a dense subgraph or on identifying an optimal clique. In these schemes, greedy approaches are implemented to find the top k-dense graph. However, the identified subgraph cannot be represented as a dense region, so the identified subgraph should be the highest-density region in the graph. In their work, Qin et al. [86] introduced a local dense subgraph (LDS) in the graph. It can be used to find the dense subregion from the subgraph and can be applied to various applications. LDSs also have some useful properties such

as being pairwise disjoint, locally dense, and compact/cohesive. The authors also presented an elegant dense subgraph model and showed that the LDS problem could be solved in polynomial time. They presented three optimization approaches to enhance the algorithm. To evaluate the LDS, they analyzed their approach using four different qualities of measures: density, relative density, edge density, and diameter. They also experimented with a real web-scale graph with 118.14 million nodes and 1.02 billion edges for demonstration of the efficiency and effectiveness of the proposed algorithm.

Zamanian et al. [87] described extensive use of a horizontal partitioning approach for the processing of a significant volume of structured data. But there is an issue in using the horizontal partitioning approach, in that the cost of the network must be minimized for the given workload and schema of the database. A popular approach to minimize the cost of the network for parallel processing of the database is copartitioning of the given tables on their join key to avoid expensive remote join operations. However, these approaches have an issue in terms of replications and copartitioning with sharing of the same join key. To resolve these issues, Zamanian et al. [87] introduced a new approach: predicate-based reference partitioning (PREF). It enables copartitioning of sets of tables on the basis of the given join predicates. The main goal of PREF is to enhance the data locality and minimize data redundancy. For this purpose, the authors also designed two new algorithms. The first algorithm requires the schema as an input, whereas the second algorithm takes the workload as an input. In their experiments, the authors showed that the workload-driven design algorithm was more efficient for a complex schema with large numbers of tables.

Chen et al. [88] noted that various existing techniques support the vertex-oriented execution model. It also allows the user to create their own logic on the vertices. There is an issue in terms of the network traffic overhead for vertex-oriented computation; however, graph partitioning is useful for minimizing the network traffic in the processing of a graph. The authors argued that very little attention has been paid to how graph partitioning can be effectively integrated into large graph processing in the cloud environment. Furthermore, they noted that the Surfer is a master–slave system with one master server and numerous slave servers (which are responsible for the graph partition and computation) and proposed a new framework of graph partitioning to enhance the performance of the network for graph partitioning itself, storage of the partitioned graph, and vertex-oriented processing of the graph. In experiments, they developed a new prototype for Pregel and enhanced it with their framework of graph partitioning. They also showed the efficiency and effectiveness of their proposed technique for the processing of large graphs.

Zeng et al. [89] pointed out that processing of distributed graphs is costly for computation of a significant volume of data in the event of moving the data among

various computers. They argued that state-of-art approaches have high computation overheads and costly communication when applied in a distributed environment. To overcome these issues, they proposed a new parallel multilevel stepwise partitioning algorithm. They divided this algorithm into two phases: an aggregate phase and a partition phase. In the first phase they used multilevel weighted label propagation for aggregation of the large graph into the small graph. It reduced the RatioCut step by step. In each step, sets of vertices were extracted by reducing part of the RatioCut, and these vertices were removed from the graph. In this way, k-way balance partitioning was obtained by use of this algorithm. In experiments, the authors performed a comparison with various other existing partitioning approaches using the data set of a graph. The algorithm performed well in comparison with state-of-the-art approaches regarding scalability and performance, and could enhance the efficiency of graph mining on a real distributed computing system.

Lee et al. [90] noted that the size and variety of information networks are growing in scientific and engineering domains these days. Because of these needs, there is another demand arising in cloud cluster computing for efficient processing of large heterogeneous graphs. The authors stated that a heterogeneous large graph has various characteristics regarding big data processing. The graph data are highly correlated, and the topological structure of a big graph can be viewed as a correlation of the vertices and edges. A graph that is heterogeneous adds an extra overhead, as compared with homogeneous graphs, regarding processing and storage. Therefore, the data generated by such random partition methods cause an extra overhead. Moreover, these types of random partition methods also cause overheads for graph patterns queries. There is another problem arising with this: how to partition graphs that perform efficiently. To resolve this problem, the authors introduced vertex block (VB) partitioning. It is a distributed model for data partitioning in large-scale graphs in the cloud. It has three features. First, it has VBs and extended VBs (EVBs) as building blocks for semantic large-scale graphs. Second, the VB partitioner uses a VB grouping algorithm to place high correlation in the graph into the same partition. Third, the VB partitioning speeds up parallel processing of graph pattern queries by minimizing interpartition query processing. In their results, the authors showed that the proposed approach has higher query latency and scalability over large-scale graphs.

LeBeane et al. [91] noted that large-scale graph analytics are an essential issue in current data centers. Large multinode processing is a critical computational problem because of the popularity of big data and cloud computing. There are many state-of-the-art frameworks available for processing of large graphs, such as PowerGraph, Pregel, and Giraph. However, there is a problem in these frameworks in that when any change is due to the cloud and big data, these frameworks cannot handle it. These state-of-the-art frameworks cannot be scalable. The data come from various sources for processing. The authors argued that the data center is trending toward a large number of heterogeneous processing nodes. Moreover, they described how a virtualized environment can create heterogeneity by partitioning the cluster of

homogeneous machines into a variety of configurations. However, the frameworks of the graph analytics are still working under assumptions. These assumptions lead to an imbalance of the load and cause faster nodes to finish processing their chunks of data earlier than slower nodes. Load balance and heterogeneous nodes become more critical for graph processing. To resolve this issue, the authors used a popular framework for heterogeneity-aware data strategies. For heterogeneous partitioning of the data, they divided the input data into shards so that every node would receive the data according to the metrics. They also defined the data-partitioning ratio between the skew factors of the clusters. In this work, they described three different types of skew factors: thread based, memory-based, and profiling-based factors. There are many complex methods available for calculation of skew factors. These can provide performance benefits for heterogeneity-aware partitioning algorithms. The authors also illustrated a simple estimate of intranode throughput in which a skew factor could drive huge performance using heterogeneity-aware partitioning strategies, and they showed that their proposed strategies reduced the execution time of the application by 64%.

Chen et al. [92] described issues and challenges in graph partitioning and computation, especially in a natural graph with skewed distribution. They argued that existing graph-processing systems have the problem of "one size fits all" with impacts on performance, load imbalance, and contention for high-degree vertices. Existing graph-processing systems also have high communication costs and high memory consumption. To overcome these issues, they introduced a new graph engine called PowerLyra. It is a hybrid and adaptive design that uses dynamic partition and computation approaches for various vertices. It combines the edge cut and vertex cut with heuristics using a hybrid algorithm of graph partitioning. It also provides data locality–aware layout optimization to enhance the cache locality during communication. The authors also experimented with using two different clusters for the graph analytics and the machine learning and data mining algorithm. The system performed much faster and consumed less memory.

Graph partitioning is a popular strategy nowadays to balance workload due to the scale of graph data and increasing availability. Because of the cost of partitioning of graphs, researchers have recently focused more on stream graph partitioning, which is very fast, updated incrementally, and easy to parallelize. However, it also has challenges such as an imbalanced workload due to access patterns during the supersets. Xu et al. [93] proposed a log-based dynamic graph partitioning method. This method uses recodes and reuses historical statistical information to refine the partitioning result. It can be used as middleware and deployed on many existing parallel graph-processing systems. It also uses historical partitioning results for creation of a hypergraph, and it uses new hypergraph streaming strategies to generate a better stream graph partitioning result. Moreover, it also dynamically partitions huge graphs and uses the system to optimize the graph partitioning to enhance the performance.

Nowadays, searching and mining of large graphs is very critical in various application domains. Processing of large scalable graphs needs careful partitioning and distribution of graphs across clusters. Yang et al. [94] explored the issue of managing large-scale clusters and various properties of local queries such as random walk, SPARQL queries, and breadth-first search. They also proposed a new approach called a self-evolving distributed graph management environment (Sedge). It reduces communication during the processing of the graph query on multiple machines. It has two levels of partitioning: primary partitioning and dynamic secondary partitioning. These two types of partitions can adapt to any real environment. The results showed that it enhanced the distributed graph processing on the commodity clusters.

19.4.3 Genome Sequence Annotation Applications of Big Data

Genome sequencing is an emerging field of research in various domains such as big data, biomedicine, and biology. The data are increasing with scientific research, but the associated technologies depend on acceleration techniques. Enhancements in genome sequencing lead to bottlenecks from the sequencing to the short-read mapping problem. The short-read problem works well under the MapReduce framework but can lead to degradation of the performance in terms of the response time.

Wang et al. [95] proposed a new architecture for acceleration of reading and MapReduce processing when it faces many requests from the field-programmable gate array (FPGA). They introduced a MapReduce framework to manage the specific tasks in the FPGA. The MapReduce algorithm is based on the restricted master problem (RMP) sequencing algorithm. The authors noted that there are three types of state-of-the-art acceleration techniques for short-read mapping, which are graphics processing unit (GPU) based, FPGA based, or MapReduce based. They evaluated the proposed techniques with respect to hardware utilization, sensitivity, the error rate, and hardware speed-up.

Jaiswal et al. [96] introduced an enhanced framework for genomics using big data computing. In this framework, they noted that genomics is all about the study of a genetic organism. Genomics includes mapping, sequencing, and analyzing of a broad range of codes of DNA, RNA, etc. The authors pointed out that with the arrival of a genome sequence, a large number of nucleotide and amino acid data sequences is formed. It is essential to verify and analyze these data effectively and efficiently, but this depends on how accurately we interpret the huge volume of high-dimensional data. There is another problem in that they increase suddenly with exponential growth. The scalability of the data tools that are currently available do not provide such analysis as is required. The authors noted that there are two types of genomics computing: cloud computing and big data computing. Presently, the main challenge is processing of data to maintain the infrastructure of growth of the data. However, people are applying a MapReduce-based technique in the cloud, with

advantages such as reducing the memory and execution time. But the cloud itself also has problems such as data protection, availability, and recovery. Qin et al. [97] described various state-of-the-art high-throughput and data collaborative techniques for biomedical and biological methods. RNA (ribonucleic acid) is also the product of DNA transcription. For the last few years, researchers have been working on new technologies with high throughput such as next-generation sequencing (NGS) and microarrays. Qin et al. [97] studied NGS, which is used to discover genetic variations linked with diseases. At present, HPCCs handle and store the sequencing data, requiring huge amounts of storage and high requirements for computation speed. Thus, processing of the data is a challenging issue. Different tools have been developed for interpretation and integration of various data types. Because of technical flaws and noise, these are not up to the mark. The main objective of biomedicine is to minimize noise and enhance the efficiency and accuracy of computation of biological processes, mathematics, statistics, and IT science.

Davis et al. [98] noted that in metabolic genome modeling, maintaining genome consistency is necessary for various computational tasks. A process is implemented to enhance the consistency of annotation among all microbial genomes of the protein. A solid cluster is fully automated because of the generation; it also provides opposite methodologies to state-of-the-art approaches to genome annotations that structure hierarchical annotations like seed subsystems. Davis et al. [98] compared their seed genome annotation with other regularly used resources such as Integrated Microbial Genomes (IMG) and RefSeq.

Yeo et al. [99] noted that big data management is typically required in healthcare and data-intensive applications. They concentrated on an application of healthcare for testing the scalability of commercial big data platforms with MapReduce. Bowtie, Contrail-bio, and basic local alignment search tool (BLAST) genomics workloads were selected and tested on a Hadoop cluster with HDFS as a file system. The results showed that the cluster could handle the extensive variety of applications of bioinformatics while providing suitable computational scalability and efficiency. By compression of the intermediate data, the process was sped up by 20%.

Heinzlreiter et al. [100] focused on genome sequence implementations and comparisons with the bioinformatics domain relying on the HBase tool running on top of Hadoop for MapReduce computation. They developed an application of the genome in Hadoop and also performed a comparison with an HBase table and execution in MapReduce. The proposed a strategy that supports reuse of intermediate data, generated before the processing step performed on a single genome. Input data for the preprocessing steps performed by the MapReduce jobs are compared with the gene. The results are stored in HBase tables, and the graph is generated for scalable genome comparison. Frequent item set mining (FIM) is an essential topic of the research because of its various applications such as DNA sequence discovery, real-world applications, and pattern-mining behaviors of humans. The FIM process is computation and memory intensive. Nowadays, since data are increasing exponentially, challenges and issues such as scalability and efficiency are becoming more severe. Liang et al. [101] proposed a distributed FIM algorithm, named Sequence-Growth, for the MapReduce framework. The proposed

algorithm constructs a tree in the lexicographical tree sequence, which permits discovery of all frequent item sets without an exhaustive search of the transaction database. The authors found that the proposed algorithm efficiently removes the generation of a large volume of intermediary data and also executes the algorithm in memory fitted in a better way. They found that their sequence growth could be modified easily according to association rules, and mining algorithms could be adapted on the MapReduce framework easily.

There have been many recent advances in technologies such as the NGS tool to enhance the speed of DNA collection of samples, sequencing, preparation, and collection. In a single run, one genetic sequence can generate a genetic sequence of over 60 GB. But the problem of recognizing the system of the human sample can take up to 45 days, which is a major bottleneck for sequence analysis and shipment of samples to a sequencing center. Dodson et al. [102] proposed a new fast and efficient method for genetic sequencing and analysis, called the dynamic distributed dimensional data model (D4M). D4M is a novelty in computer programming, which associates the characteristics of five processing technologies: an associative array, sparse linear algebra, a triple-store database, linear algebra, and a distributed array. In the base triple-store database, large volumes of data are handled. D4M provides parallelism by using linear algebra on the interfaces of the triple store. There is a huge challenge in identifying the gene sequence repetitively that occurs within the DNA sequence. It is very simple conceptually, but computationally it is a huge challenge. Moreover, there is also a big challenge for biological research in that many regions within the sequence of genomics have not changed for many years. Identifying these uncovered regions is a big challenge for researchers. The main problem arises in analysis of a significant volume of data. It increases with each genome that is sequenced. State-of-the-art approaches need pairwise sequences for comparison of chromosomes, which consume a lot in execution. Phinney et al. [103] designed a new algorithm that partitions the sequence of the genome on the basis of the value and the repetitive sequence. It consists of a single aggregation of global steps that recognize all matches among the sequence concurrently. The scalability of this approach is its main characteristic. Additionally, using more nodes for computation can increase the performance of the system.

Toh et al. [104] proposed a sequence search technique and alignment by using a sequence search and alignment by hashing algorithm (SSAHA) for searching a sequence from a database. In this approach, they divided each database into k-tuples of k-contiguous bases and then processed the query base by base. They found that if a query did not match on the first hit, there was a strong possibility that it would not match in later hits. They argued that it is possible to apply BLAST on the first stage of the SSAHA, so it can create overlapping words with consistent length of a query that can increase the sensitivity. Therefore, they also proposed a new technique called the basic sequence search by hash algorithm (BSSHA). It divide the algorithms into two parts: a time hash for creating the hash table in the database, and a time search for the processing of the query. Moreover, a database table is created in the main memory; it resides in the memory so it will save the time

of disk access. The authors also showed that the time complexity and processing time for the BSSHA are less than those for existing techniques such as the SSAHA.

Meng et al. [105] noted that for processing of bioinformatics information, sequence alignment is the basic method and information evaluation. They stated that bioinformatics can be divided into two categories. In the first category there are two types: one is pairwise sequence alignment and the second is multiple sequence alignment. In the second category there are also two types: one is local alignment based on the sequence range and the second is global alignment based on the whole sequence alignment. A local sequence is much more useful than global alignment. Currently, serial BLAST is very complex and not useful for a large volume of data. It does not meet the needs of the current large volume of genetic data; implantation on a GPU and heterogeneous computing is very complicated.

As there is exponential growth in the database of the genome, it is tough to process these biological computations. O'Driscoll et al. [106] introduced a new approach, Hadoop BLAST (HBlast). This scheme partitions the sequence of the query by using virtual partitioning. The proposed method improves scalability in comparison with other available solutions and also balances the computation workload. BLAST is very commonly used for bioinformatics programs to search the available sequence for similarities among DNA and proteins, using the technique of sequence alignment. Sait et al. [107] evaluated the performance of serial and parallel BLAST using the traditional state of a diskless HPCC. It enhanced the reliability and minimized the cost of the communications as compared with the traditional diskfull clusters.

Boratyn et al. [108] developed a tool called context-sensitive blast (CS-BLAST), which combines information from recently searched queries derived from the protein profile to achieve better homology detection than position-specific iterated BLAST (PSI-BLAST), which constructs a position-specific matrix from scratch. They also proposed a new method called domain enhancement lookup time accelerated BLAST (DELTA-BLAST). It begins with a query from the conserved domain database, then it makes multiple alignments of the conserved domains after it computes the position-specific score matrix (PSSM); finally, it applies a sequence search query to the database.

Table 19.1 provides a taxonomy of relevant research efforts in healthcare and smart city applications, graph-based applications, and genome sequence annotation applications. Table 19.2 provides a taxonomy of numerous tools and technologies, along with their salient features and limitations.

19.5 Conclusion

In this chapter we have critically analyzed some of the core applications of big data and their impacts in improving the quality of human life by primarily focusing on healthcare and smart city applications, genome sequence annotation applications, and graph-based applications. The volumes of structured, semistructured, and

Table 19.1 Big data applications: taxonomy of research efforts

Applications of big data	Research efforts
Healthcare and smart city	Smart city, challenges and issues, data analytics (Nuaimi et al. [51]); healthcare, healthcare tools, graph based; e-health, secure integration, IoT ([45]); healthcare, strengths, weaknesses, threats, and opportunities (Collins [53]); data analytics, healthcare, architecture, tools of big data (Raghupathi and Raghupathi [54]); smart city, transport, capability sharing, optimization (Mehmood and Graham [8]); healthcare, analysis, steps for big data projects (Huang et al. [55]); information retrieval, storage, and capture issues, patient similarity (Barkhordari et al. [56]); big data challenges, incompleteness, heterogeneity framework (Toga et al. [57])
Graph based	Data placement, data locality, and DGM (Wang et al. [72]); graph analytics, mining algorithms (Lee et al. [73]); data analytics, graph processing, data locality (Xia et al. [74]); tNMF, BRDG (Fang et al. [76]); graph based, scheduling, data locality, global optimization (Xue et al. [77])
Genome sequence annotation	NGS, FPGA, short-read problems (Wang et al. [95]); BLAST, genomics computing issues (Jaiswal et al. [96]); biomedical and biological methods, biomedicine, NGS (Qin et al. [97]); sequence annotation, genome, hierarchical annotations (Davis et al. [98]); Bowtie, Contrail-bio, BLAST, evaluation, Hadoop (Yeo et al. [99]); NGS, genome (Heinzlreiter et al. [100]); FIM, exhaustive search, tree-based algorithm (Liang et al. [101]); D4M, challenges and issues of DNA sequence (Dodson et al. [102]); pairwise sequence, overhead, partitioning scheme, HBase (Phinney et al. [103]); SSAHA, BLAST, BSSHA (Toh et al. [104]); bioinformatics, pairwise and multiple sequence alignment, local alignment (Meng et al. [105]); HBlast (O'Driscoll et al. [106]); BLAST, serial and parallel BLAST (Sait et al. [107]); DELTA-BLAST (Boratyn et al. [108])

BLAST basic local alignment search tool, *BRDG* bipartite request dependency graph, *BSSHA* basic sequence search by hash algorithm, *D4M* dynamic distributed dimensional data model, *DELTA* domain enhancement lookup time accelerated, *DGM* data group matrix, *FIM* frequent item set mining, *FPGA* field-programmable gate array, *HBlast* Hadoop BLAST, *IoT* internet of things, *NGS* next-generation sequencing, *SSAHA* sequence search and alignment by hashing algorithm, *tNMF* tri-nonnegative matrix factor

unstructured data generated from heterogeneous sources is increasing exponentially and brings many challenges. Numerous tools and technologies have been developed over the years to address these issues. This chapter has also presented a summary

Table 19.2 Big data tools and technologies

Tool	Type	Layers	Features	Limitations
Hadoop [15]	Open-source platform	Management, processing, storage	Scalability, high availability, fault tolerance, reliability, storage, low-cost, distributed, local processing, parallel computing, cost effective	In-memory computation for iterative algorithms, I/O disk bottleneck, poor resource utilization, not optimized for iterative and interactive applications
HDFS [12]	Open-source platform	Storage	Scalable, reliable, fault tolerance, operability, high availability, reduces network I/O, task scheduling	HDFS is immutable, high cost due to replications, scalability issues as local data increase
YARN [26]	Open-source platform	Management	Job scheduling, cluster resource management, multitenancy, compatibility, scalability, cluster utilization	Does not support short interactive jobs or long-running services
MapReduce [13]	Open-source platform	Processing	Parallel processing, data-partitioning map, reduce, faster processing	Performance issues, extensions of programming models, needs more data-aware optimizations
Pig [16]	Open-source platform	Processing, analysis	Extensibility, ease of programming, optimization opportunities, self-optimizing, processing of data iteratively, pipeline of ETL data	Implicit data schema, not mature, lack of technical support
Hive [17]	Open-source platform	Management	Ad hoc queries, analysis of huge data sets, summarization of stored data, quick response over large data set	High latency, performance, complexity, complicated data updates, no access to real-time data
Storm [18]	Open-source platform	Processing	Reliable, easy to operate, real-time analytics, scalable, fault tolerant, continuous computation, fast processing, distributed RPC, ETL	Requires more advanced configurations, cannot integrate with storage engine directly
Spark [22]	Open-source platform	Management, processing, storage, analysis	Scalability, fault tolerance, in-memory computation, parallel computing, support for genome sequences, ease of use, fast processing, unified engine, local processing, support for multiple languages, advanced analytics, RDD, simplicity	Large system resources, memory consumption issues, high cost, not mature

(continued)

Table 19.2 (continued)

Tool	Type	Layers	Features	Limitations
Spark Streaming [22]	Open-source platform	Processing	Fault tolerant, scalable, high throughput, high-level functions, Spark integration, ease of use with various programming languages	Streaming processing has additional extra latency, data loss can happen when a working node fails
BlinkDB [23]	Open source	Processing	Interactive, fast and reliable processing, bootstrap performance diagnostics, accuracy of query, large query processing	It executes the newly created sample module offline, response time is not good, also has an issue with approximation query processing
BDAS [24]	Open-source platform	Processing, analysis	Resource virtualization and utilization, different layers such as storage, processing engine, access and interfaces	Depends heavily on storage of data in the memory, making the whole stack hard to maintain and integrate
Mesos [25]	Open-source platform	Management	Scalability, fault tolerance, scheduling of multiple resources such as ports, memory, disk, and CPU	No support for Kerberos, DEA has no data locality, inefficient cluster utilization, limitation of one executor per slave
MongoDB [33]	Open-source platform	Management, processing, storage, analysis	Automatic scaling, high performance, high availability, scalability, aggregation, load balancing, native replication, security, rich query language	No joins available in MongoDB, problem in the case of multiple queries resulting in inflexibility; it consumes more memory and also has some concurrency issues
R [34]	Open-source platform	Analysis	Statistical analysis, packages, graphical visualization, classification, classical data testing, clustering, analysis of time series, graphical methods, nonlinear and linear modeling	Parallelization complexity, lack of graphical user interface intimidating for beginners
Drayed [35]	Open source	Processing	Flexible, parallel processing, long-term preservation, promotes data visibility, efficient utilization of resources	Inexperienced programmers, lack of effective data analysis support
Syncfusion [42]	Commercial	Management, processing, storage, analysis	Query processing on structured and unstructured data, cost-effective storage, linear scalability	Issues with layout process, RDL format; also has unsupported expression

Cloudera [43]	Commercial platform	Processing	Flexibility, integration, security, scalability, high availability and compatibility, provides Impala, reliable model, secure, simple, distributed computing, web-based interface with enterprise capabilities and scalable storage	Difficulties with learning and configuration
Pivotal HD [45]	Commercial platform	Analysis	Flexible models, backup and recovery, data protection, georedundancy, robust availability in case of failure	Pivot table creation is very complex; it also has a data concurrency problem; performance issue with large volume of data; manual calculation of metrics is time consuming
EC2 [46]	Commercial platform	Processing, storage	Flexibility, security, scalability, reliability, fault tolerance	Very costly, unaffordable for common users
Neo4j [38]	Open source	Processing, storage	Graph-based storage, graph-based processing engine, query processing, scalability, vertical scaling, higher performance and concurrency, support for various languages	No support for sharing; limitations on node properties and relationships
Pentaho [39]	Open source	Analysis	ETL, BI, data integration, dashboarding capabilities, OLAP services, data mining, data analytics, visualization, no coding required for integration	Data visualization and analytics tool requires more improvement; it is inconsistent and inconvenient; the layer of metadata is very hard to use and understand; moreover, there is little help from the documentation
Talend [40]	Open source	Processing, management	Scalability, ease of use, reliability, integration, ETL, productivity, data management	There is a need for JDBC for resource access; no product for metadata management and quality of data; bottleneck in jobs due to automation of data partitioning and repartitioning, and allocation of resources across the grid

(continued)

Table 19.2 (continued)

Tool	Type	Layers	Features	Limitations
Tableau [47]	Commercial platform	Analysis	Fast and interactive reports, excellent user interface, integration, mobile support, customer services, user forum, low-cost, easy to upgrade and use	Initial data, statistical features are avoided in this framework and financial reporting
IBM [48]	Commercial platform	Processing, management, storage, analysis	Data mining, data integration, BI computing of stream, solution of data warehouse and Enterprise Class Hadoop; dynamically changes real-time data	Poor data analytics and lack of support for understanding the quality of data; absence of business cases; granularity of big data is overlooked
SAS [49]	Commercial platform	Analysis	Provides high-performance analytics, grid commuting, database analytics, in-memory computation; solution of complex problems; advanced analytical tool	Need to understand the graph package for customization of plots; it is expensive and not attainable for every professional, so it is very difficult to access
Oracle [41]	Open source	Storage, processing, analysis	Advanced analytics, in-memory computation, scalability, batch facilities, faster development, data integrator, environment for statistical computing	Lack of support to handle large volumes of data
Teradata's Aster [50]	Commercial platform	Analysis	It has rough methods to deal with large volumes of data (i.e., full scanning); it is computationally similar to Hadoop	Teradata is not a mature product and has limited scalability

BDAS Berkeley Data Analytics Stack, *BI* business intelligence, *CPU* central processing unit, *DEA* dynamic executor allocation, *ETL* extraction, transformation, and loading, *HDFS* Hadoop File System, *JDBC* Java Database Connectivity, *OLAP* online analytical processing, *RDD* resilient distributed data sets, *RDL* Report Definition Language, *RPC* remote procedure call

of state-of-the-art tools and methodologies for processing of big data applications. This overview has included a discussion about the objectives, methodologies, and key approaches in the existing techniques. A taxonomy of the research efforts within each application domain has also been presented.

The issues related to big data are immense and cover a variety of challenges in different phases of big data classification, i.e., data acquisition, data transportation, data storage, compression/decompression, preprocessing, data analysis, and data visualization. These challenges demand collaborative efforts from multidisciplinary domains to obtain insights into a wealth of information to make informative decisions to improve the quality of human life.

Acknowledgements The authors acknowledge with thanks the technical and financial support received from the Deanship of Scientific Research (DSR) at the King Abdulaziz University (KAU), Jeddah, Saudi Arabia, under grant number G-651-611-38. The work carried out in this paper is supported by the High Performance Computing (HPC) Center at the King Abdulaziz University.

References

1. Mehmood, R., Faisal, M.A., Altowaijri, S.: Future networked healthcare systems: a review and case study. In: Boucadair, M., Jacquenet, C. (eds.) Handbook of Research on Redesigning the Future of Internet Architectures, pp. 531–558. IGI Global, Hershey, PA (2015)
2. Usman, S., Mehmood, R., Katib, I.: Big data and HPC convergence: the cutting edge and outlook. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 11–26. Springer, Cham (2018)
3. Alam, F., Mehmood, R., Katib, I., Albogami, N.N., Albeshri, A.: Data fusion and IoT for smart ubiquitous environments: a survey. IEEE Access. **5**, 9533–9554 (2017)
4. Muhammed, T., Mehmood, R., Albeshri, A., Katib, I.: UbeHealth: a personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities. IEEE Access. **6**, 32258–32285 (2018)
5. Suma, S., Mehmood, R., Albugami, N., Katib, I., Albeshri, A.: Enabling next generation logistics and planning for smarter societies. Procedia Comput. Sci. **109**, 1122–1127 (2017)
6. Suma, S., Mehmood, R., Albeshri, A.: Automatic event detection in smart cities using big data analytics. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 111–122. Springer, Cham (2018)
7. Mehmood, R., Alam, F., Albogami, N.N., Katib, I., Albeshri, A., Altowaijri, S.M.: UTiLearn: a personalised ubiquitous teaching and learning system for smart societies. IEEE Access. **5**, 2615–2635 (2017)
8. Mehmood, R., Graham, G.: Big data logistics: a health-care transport capacity sharing model. Procedia Comput. Sci. **64**, 1107–1114 (2015)
9. Yadranjiaghdam, B., Pool, N., Tabrizi, N.: A survey on real-time big data analytics: applications and tools. In: 2016 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 404–409. IEEE, Piscataway (2016)
10. Oussous, A., Benjelloun, F.-Z., Ait Lahcen, A., Belfkih, S.: Big data technologies: a survey. J. King Saud Univ. Comput. Inf. Sci. **30**, 431–448 (2018)

11. Philip Chen, C.L., Zhang, C.-Y.: Data-intensive applications, challenges, techniques and technologies: a survey on big data. Inf. Sci. **275**, 314–347 (2014)
12. Borthakur, D.: HDFS architecture guide. Apache Software Foundation. https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html. Accessed
13. Dean, J., Ghemawat, S.: MapReduce. Commun. ACM. **51**, 107 (2008)
14. Vavilapalli, V.K., Seth, S., Saha, B., Curino, C., O'Malley, O., Radia, S., Reed, B., Baldeschwieler, E., Murthy, A.C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H.: Apache Hadoop YARN. In: Proceedings of the 4th Annual Symposium on Cloud Computing—SOCC'13, pp. 1–16. ACM, New York (2013)
15. Apache Hadoop 2.9.1—Apache Hadoop YARN. Apache Software Foundation. http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html. Accessed
16. Welcome to Apache Pig! Apache Software Foundation. https://pig.apache.org/. Accessed
17. Apache Hive™. Apache Software Foundation. http://hive.apache.org/. Accessed
18. Apache Storm. Apache Software Foundation. http://storm.apache.org/. Accessed
19. Apache Spark™—unified analytics engine for big data. Apache Software Foundation. https://spark.apache.org/. Accessed
20. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. ACM Digital Library (2010). https://dl.acm.org/citation.cfm?id=1863103.1863113. Accessed 4 Aug 2018
21. Spark Streaming|Apache Spark. Apache Software Foundation. http://spark.apache.org/streaming/. Accessed 4 Aug 2018
22. Apache Spark: Apache Spark™—Lightning-Fast Cluster Computing
23. Agarwal, S., Panda, A., Mozafari, B., Madden, S., Stoica, I., Panda, A., Milner, H., Madden, S., Stoica, I., Mozafari, B., Madden, S., Stoica, I., Berkeley, U.C.: BlinkDB: queries with bounded errors and bounded response times on very large data. Proceedings of ACM EuroSys 2013, Prague
24. Software. AMPLab—UC Berkeley. https://amplab.cs.berkeley.edu/software/. Accessed 5 Aug 2018
25. Hindman, B.: Apache Mesos. Apache Software Foundation. http://mesos.apache.org/. Accessed 5 Aug 2018
26. Murthy, A.C.: Apache Hadoop YARN. Apache Software Foundation. http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html. Accessed 5 Aug 2018
27. Li, H., Ghodsi, A., Zaharia, M., Baldeschwieler, E., Shenker, S., Stoica, I.: Tachyon: memory throughput I/O for cluster computing frameworks. Memory. **18**, 1 (2013)
28. ML Base. AMP Lab, UC Berkeley. http://www.mlbase.org. Accessed 5 Aug 2018
29. MLlib | Apache Spark
30. Spark SQL & DataFrames | Apache Spark SQL. Apache Software Foundation. https://spark.apache.org/sql/. Accessed 6 Aug 2018
31. GraphX | Apache Spark. Apache Software Foundation. http://spark.apache.org/graphx/. Accessed 5 Aug 2018
32. Xin, R.S., Gonzalez, J.E., Franklin, M.J., Stoica, I.: GraphX. In: First International Workshop on Graph Data Management Experiences and Systems—GRADES'13, pp. 1–6. ACM, New York (2013)
33. MongoDB for GIANT Ideas. MongoDB Inc. https://www.mongodb.com/. Accessed 5 Aug 2018
34. Chambers, J.: Bell Laboratories: What is R? The R Foundation. http://www.r-project.org/. Accessed 5 Aug 2018
35. Data Dryad. http://datadryad.org. Accessed 5 Aug 2018
36. NexisNexis Risk Solutions. HPCC Systems. http://hpccsystems.com. Accessed 5 Aug 2018
37. Sagiroglu, S., Sinanc, D.: Big data: a review. Int. Conf. Collab. Technol. Syst. 42–47 (2013)
38. Neo4j: the world's leading graph database. Neo4j, Inc. http://neo4j.com. Accessed 7 Aug 2018

39. Pentaho | Data Integration, Business Analytics and Big Data Leaders. http://www.pentaho.com. Accessed 7 Aug 2018
40. Talend open source data integration software. https://www.talend.com. Accessed 7 Aug 2018
41. Big data | what is big data? Oracle. https://www.oracle.com/big-data/index.html. Accessed 7 Aug 2018
42. Syncfusion Big Data Platform | Big Data Platform simplifies working with Hadoop on Windows. https://www.syncfusion.com/products/big-data. Accessed 7 Aug 2018
43. Cloudera. http://www.cloudera.com. Accessed 7 Aug 2018
44. Cloudera. Implementing active/active multi-cluster deployments with Cloudera Enterprise. http://www.cloudera.com/content/dam/www/static/documents/whitepapers/active-active-deployments-with-cloudera-enterprise-whitepaper.pdf. Accessed 7 Aug 2018
45. Pivotal HDB | Big Data (2015)
46. Amazon Web Services: Amazon Web Services (AWS)—Cloud Computing Services. http://aws.amazon.com. Accessed 7 Aug 2018
47. Business Intelligence and Analytics | Tableau Software. http://www.tableau.com. Accessed 7 Aug 2018
48. IBM big data platform—Bringing big data to the Enterprise (2016)
49. Big Data & IoT insights. SAS. http://www.sas.com/en_us/insights/big-data.html. Accessed 5 Aug 2018
50. Big data, big data beyond the hype and big data successes. Teradata. http://bigdata.teradata.com/. Accessed 5 Aug 2018
51. Nuaimi, E.A., Neyadi, H.A., Mohamed, N., Al-Jaroodi, J.: Applications of big data to smart cities. J. Internet Serv. Appl. **6**, 25 (2015)
52. Suciu, G., Suciu, V., Martian, A., Craciunescu, R., Vulpe, A., Marcu, I., Halunga, S., Fratu, O.: Big data, internet of things and cloud convergence—an architecture for secure E-health applications. J. Med. Syst. **39**, 141 (2015)
53. Collins, B.: Big data and health economics: strengths, weaknesses, opportunities and threats. PharmacoEconomics. **34**, 101–106 (2015)
54. Raghupathi, W., Raghupathi, V.: Big data analytics in healthcare: promise and potential. Heal. Inf. Sci. Syst. **2**, 3 (2014)
55. Huang, T., Lan, L., Fang, X., An, P., Min, J., Wang, F.: Promises and challenges of big data computing in health sciences. Big Data Res. **2**, 2–11 (2015)
56. Barkhordari, M., Niamanesh, M.: ScaDiPaSi: an effective scalable and distributable MapReduce-based method to find patient similarity on huge healthcare networks. Big Data Res. **2**, 19–27 (2015)
57. Toga, A.W., Dinov, I.D.: Sharing big biomedical data. J. Big Data. **2**, 7 (2015)
58. Ahmed, W., Khan, M., Khan, A.A., Mehmood, R., Algarni, A., Albeshri, A., Katib, I.: A framework for faster porting of scientific applications between heterogeneous clouds. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, pp. 27–43. Springer, Cham (2018)
59. Alotaibi, S., Mehmood, R.: Big data enabled healthcare supply chain management: opportunities and challenges. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 207–215. Springer, Cham (2018)
60. Alamoudi, E., Mehmood, R., Albeshri, A., Gojobori, T.: DNA profiling methods and tools: a review. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 216–231. Springer, Cham (2018)

61. Al Shehri, W., Mehmood, R., Alayyaf, H.: A smart pain management system using big data computing. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 232–246. Springer, Cham (2018)
62. Khanum, A., Alvi, A., Mehmood, R.: Towards a semantically enriched computational intelligence (SECI) framework for smart farming. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 247–257. Springer, Cham (2018)
63. Aqib, M., Mehmood, R., Albeshri, A., Alzahrani, A.: Disaster management in smart cities by forecasting traffic plan using deep learning and GPUs. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 139–154. Springer, Cham (2018)
64. Alam, F., Mehmood, R., Katib, I.: D2TFRS: an object recognition method for autonomous vehicles based on RGB and spatial values of pixels. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 155–168. Springer, Cham (2018)
65. Muhammed, T., Mehmood, R., Albeshri, A.: Enabling reliable and resilient IoT based smart city applications. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 169–184. Springer, Cham (2018)
66. Al-Dhubhani, R., Mehmood, R., Katib, I., Algarni, A.: Location privacy in smart cities era. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 123–138. Springer, Cham (2018)
67. Alomari, E., Mehmood, R.: Analysis of tweets in Arabic language for detection of road traffic conditions. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 98–110. Springer, Cham (2018)
68. Arfat, Y., Aqib, M., Mehmood, R., Albeshri, A., Katib, I., Albogami, N., Alzahrani, A.: Enabling smarter societies through mobile big data fogs and clouds. Procedia Comput. Sci. **109**, 1128–1133 (2017)
69. Schlingensiepen, J., Nemtanu, F., Mehmood, R., McCluskey, L.: Autonomic transport management systems—enabler for smart cities, personalized medicine, participation and industry grid/industry 4.0. In: Intelligent Transportation Systems—Problems and Perspectives, pp. 3–35. Springer, Cham (2016)
70. Alyahya, H., Mehmood, R., Katib, I.: Parallel sparse matrix vector multiplication on Intel MIC: performance analysis. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 306–322. Springer, Cham (2018)
71. Arfat, Y., Mehmood, R., Albeshri, A.: Parallel shortest path graph computations of United States road network data on Apache Spark. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 323–336. Springer, Cham (2018)

72. Wang, J., Xiao, Q., Yin, J., Shang, P.: DRAW: a new data-grouping-aware data placement scheme for data intensive applications with interest locality. IEEE Trans. Magn. **49**, 2514–2520 (2013)
73. Lee, S., Sukumar, S.R., Hong, S., Lim, S.-H.: Enabling graph mining in RDF triplestores using SPARQL for holistic in-situ graph analysis. Expert Syst. Appl. **48**, 9–25 (2016)
74. Xia, Y., Tanase, I.G., Nai, L., Tan, W., Liu, Y., Crawford, J., Lin, C.: Explore efficient data organization for large scale graph analytics and storage. In: Proceedings of the 2014 IEEE BigData Conference, pp. 942–951 (2014)
75. Malewicz, G., Austern, M.H., Bik, A.J., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: a system for large-scale graph processing. In: Proceedings of the 2010 International Conference on Management of Data—SIGMOD'10, pp. 135–146 (2010)
76. Fang, C., Secondary, C.A., Author, C., Fang, C., Liu, J., Ansari, N., Fang, C.: Wireless networks revealing connectivity structural patterns among web objects based on co-clustering of bipartite request dependency graph revealing connectivity structural patterns among web objects based on co-clustering of bipartite request dependency. Under Rev
77. Xue, R., Gao, S., Ao, L., Guan, Z.: BOLAS: bipartite-graph oriented locality-aware scheduling for MapReduce tasks. In: 2015 14th International Symposium on Parallel and Distributed Computing, pp. 37–45. IEEE, Piscataway (2015)
78. Orozco, D., Garcia, E., Gao, G.: Locality optimization of stencil applications using data dependency graphs. In: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 6548 LNCS, pp. 77–91 (2011)
79. Hassanzadeh-Nazarabadi, Y., Küpçü, A., Özkasap, Ö.: Locality aware skip graph. In: Proceedings of the 2015 IEEE 35th International Conference on Distributed Computing Systems Workshops. ICDCSW 2015, pp. 105–111 (2015)
80. Kandemir, M., Choudhary, A., Ramanujam, J., Banerjee, P.: A graph based framework to detect optimal memory layouts for improving data locality. In: Proceedings of the 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing, San Juan (1999). doi: 10.1109/IPPS.1999.760558
81. Chernov, A., Belevantsev, A., Malikov, O.: A thread partitioning algorithm for data locality improvement. In: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 3019, pp. 278–285 (2004)
82. Zhang, Y.M., Huang, K., Geng, G., Liu, C.L.: Fast kNN graph construction with locality sensitive hashing. In: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 8189 LNAI, pp. 660–674 (2013)
83. Zhang, M., Shen, F., Zhang, H., Xie, N., Yang, W.: Fast graph similarity search via locality sensitive hashing. Adv. Multimed. Inf. Process. PCM 2015. **9315**, 447–455 (2015)
84. Yuan, P., Xie, C., Liu, L., Jin, H., Member, S.: PathGraph: a path centric graph processing system. IEEE Trans. Parallel Distrib. Syst. **9219**, 1–15 (2016)
85. Shao, Y., Cui, B., Ma, L.: PAGE: a partition aware engine for parallel graph computation. IEEE Trans. Knowl. Data Eng. **27**, 518–530 (2015)
86. Qin, L., Li, R.-H., Chang, L., Zhang, C.: Locally densest subgraph discovery. In: Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining—KDD'15, pp. 965–974 (2015)
87. Zamanian, E., Binnig, C., Salama, A.: Locality-aware partitioning in parallel database systems. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 17–30 (2015)
88. Chen, R., Yang, M., Weng, X., Choi, B., He, B., Li, X.: Improving large graph processing on partitioned graphs in the cloud. In: Proceedings of the Third ACM Symposium on Cloud Computing—SoCC'12, pp. 1–13 (2012)

89. Zeng, Z., Wu, B., Wang, H.: A parallel graph partitioning algorithm to speed up the large-scale distributed graph mining. In: Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications—BigMine'12, pp. 61–68 (2012)
90. Lee, K., Liu, L.: Efficient data partitioning model for heterogeneous graphs in the cloud. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage Analysis, pp. 1–12 (2013)
91. LeBeane, M., Song, S., Panda, R., Ryoo, J.H., John, L.K.: Data partitioning strategies for graph workloads on heterogeneous clusters. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage Analysis—SC'15, pp. 1–12 (2015)
92. Chen, R., Shi, J., Chen, Y., Chen, H.: PowerLyra: differentiated graph computation and partitioning on skewed graphs. In: Proceedings of the Tenth European Conference on Computer Systems—EuroSys'15, pp. 1–15 (2015)
93. Xu, N., Chen, L., Cui, B.: LogGP: a log-based dynamic graph partitioning method. Proc. VLDB Endow. **7**, 1917–1928 (2014)
94. Yang, S., Yan, X., Zong, B., Khan, A.: Towards effective partition management for large graphs. In: Proceedings of the 2012 International Conference on Management Data—SIGMOD'12, pp. 517–528 (2012)
95. Wang, C., Li, X., Chen, P., Wang, A., Zhou, X., Yu, H.: Heterogeneous cloud framework for big data genome sequencing. IEEE/ACM Trans. Comput. Biol. Bioinforma. **12**, 166–178 (2015)
96. Jaiswal, A., Upadhyay, A.: An Enhanced Framework of Genomics Using Big Data Computing. Proceedings of the 2015 International Conference on Computer, Communication and Control (IC4), Indore (2015)
97. Qin, Y., Yalamanchili, H.K., Qin, J., Yan, B., Wang, J.: The current status and challenges in computational analysis of genomic big data. Big Data Res. **2**, 12–18 (2015)
98. Davis, J., Olsen, G., Overbeek, R., Vonstein, V., Xia, F.: In search of genome annotation consistency: solid gene clusters and how to use them. 3 Biotech. **4**(3), 331–335 (2014)
99. Yeo, H., Crawford, C.H.: Big Data: Cloud Computing in Genomics Applications. Proceedings of the 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, 2904–2906 (2015)
100. Heinzlreiter, P., Krieger, M.T., Leitner, I.: Hadoop-based genome comparisons. In: Proceedings of the 2nd International Conference on Cloud and Green Computing, pp. 695–701 (2012)
101. Liang, Y.-H., Wu, S.-Y.: Sequence-growth: a scalable and effective frequent itemset mining algorithm for big data based on MapReduce framework. In: 2015 IEEE Int. Congr. Big Data, pp. 393–400 (2015)
102. Dodson, S., Ricke, D.O., Kepner, J.: Genetic sequence matching using D4M big data approaches. In: 2014 IEEE High Performance Extreme Computing Conference HPEC 2014 (2014)
103. Phinney, M., Cao, H., Dhroso, A., Shyu, C.: Ecosystem. pp. 4–6
104. Toh, S.-H.T.S.-H., Lee, H.-J.L.H.-J., Do, K.-H.D.K.-H.: Basic sequence search by hashing algorithm in DNA sequence databases. In: 2009 11th International Conference on Advanced Communication Technologies, pp. 2317–2320 (2009)
105. Meng, M., Gao, J., Chen, J.J.: Blast-Parallel: The parallelizing implementation of sequence alignment algorithms based on Hadoop platform. In: Proceedings of the 2013 6th International Conference on BioMedical Engineering and Informatics, BMEI 2013, pp. 465–470 (2013)
106. O'Driscoll, A., Belogrudov, V., Carroll, J., Kropp, K., Walsh, P., Ghazal, P., Sleator, R.D.: HBLAST: parallelised sequence similarity—a Hadoop MapReducable basic local alignment search tool. J. Biomed. Inform. **54**, 58–64 (2015)
107. Sait, S.M., Al-Mulhem, M., Al-Shaikh, R.: Evaluating BLAST runtime using NAS-based high performance clusters. In: Proceedings—CIMSim 2011 3rd International Conference on Computational Intelligence, Modelling and Simulation, pp. 51–56 (2011)
108. Boratyn, G.M., Schäffer, A.A., Agarwala, R., Altschul, S.F., Lipman, D.J., Madden, T.L.: Domain enhanced lookup time accelerated BLAST. Biol. Direct. **7**, 12 (2012)

Chapter 20
Big Data for Smart Infrastructure Design: Opportunities and Challenges

Yasir Arfat, Sardar Usman, Rashid Mehmood, and Iyad Katib

20.1 Introduction

Big data is a buzzword, which catches lots of attention in the recent years. It means massive amount of structured, semi-structured, and unstructured data collected from different resources and is not possible to store and process this data by traditional databases and software techniques [1]. Big data refers to the emerging technologies that are designed to extract value from data having four Vs characteristics, volume, variety, velocity, and veracity [2]. Volume refers to the vast amount of data generated every second. Just think of all the emails, machine generated data, transactions, IoT, and social media data that we share every second. When this data reaches to zettabyte or exabyte in scale, data owners need a distributed processing and storage, which does not come cheaply. Velocity refers to the speed at which new data is generated and the speed at which data moves around. As the volume and speed of data generation is increasing rapidly, there is growing need of advancing the capabilities of the devices, equipment, and software collecting data. Variety refers to the different types of data that we can use. This data comes from different resources and is heterogeneous, unstructured, and inconsistent that demands high storage capacity and efficient processing resources. Value is the most important aspect of big data; it refers to the process of discovering huge hidden values from large datasets with various types and rapid generation. Big data allows us to take informed decision from structured or unstructured data as it contains information,

Y. Arfat · S. Usman (✉) · I. Katib
Department of Computer Science, FCIT, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: yqasim@stu.kau.edu.sa; susman@stu.kau.edu.sa; iakatib@kau.edu.sa

R. Mehmood
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: RMehmood@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_20

which cannot be available in small sized data. Big data technologies are being used in many application areas, see e.g., [3–8].

Most of the existing data mining algorithms and knowledge discovery tools are optimized for structured data and is difficult to use them to extract the value of the data from unstructured or semi-structured data. Big data veracity refers to the biases, noise, and abnormality in data. Big data with their ability to extract meaningful and valuable information from huge volume of data to enhance decision-making has recently attracted lots of attention from both academia and industry. Many organizations have adopted big data solutions to improve operational efficiency, revenues and to take competitive advantage over their counterparts. The increasing demands of computational resources and HPDA (High Performance Data Analytics) resulting in use of HPC (High-Performance Computing) resources for big data solutions and vice versa open up a new dimension of converged HPC and big data environment. Smart cities and IoT (Internet of Things) are some of the emerging market segments of HPDA and different solutions starting to emerge in the recent years, such as in smart infrastructure [1, 9], healthcare [10–13], transport [6, 14–20], and other applications [21, 22].

Numerous research publications have been published over the years to give a comprehensive review of big data tools, techniques, analytics, challenges, and issues. For example, Kruse et al. [23] focused on the challenges and issues faced by big data analytics in healthcare. Sivarajah et al. [24] presented a holistic view of big data challenges and Chauhan et al. [25] focused on big data challenges and issues in smart cities. Chen et al. [26] examined the background, state-of-the-art techniques, and some applications of big data to get better understanding of big data concept in general. Our work is mainly focused on challenges and issues of big data (Hadoop), primarily focusing on data locality, scheduling, load balancing, heterogeneity, I/O issues, etc. by analyzing different research efforts over the years to address above-mentioned challenges. We also provide the taxonomy of the related research.

The rest of the chapter is organized as follows. The next section provides an abstract overview of Hadoop and its two main components. Section 20.3 gives an overview of challenges and opportunities with big data in general. Sections 20.4–20.10 provides a detailed survey of opportunities, challenges, and issues of big data (Hadoop) in terms of data locality, load balancing, heterogeneity issues, scheduling issues, in-memory computation, multiple query optimization, and I/O issues, respectively. We also provide the taxonomy of the research efforts conducted over the years to overcome these challenges. Chapter is concluded in the final section.

20.2 Hadoop

Hadoop is open source software framework that provides reliable distributed data storage with design that facilitates high scalability from single server to multiple commodity hardware, for the distributed processing of large-scale datasets

using Map-Reduce programming model. Hadoop is widely accepted and adopted successfully by many organizations like Amazon, AOL, EBay, Facebook, Twitter, etc. Hadoop has two main units, i.e., Hadoop distributed file system (HDFS) and Map-Reduce processing.

20.2.1 Hadoop Distributed File System

Hadoop distributed file system (HDFS) is a system that allows multiple commodity machines to store data from a single source. HDFS consists of name node and data node and is operated as master slave architecture. Name node serves as a master component and data nodes serve as slave components [27] (Fig. 20.1).

Name nodes comprise only the metadata information of HDFS [27, 28], i.e.,

- The blocks of data that are present on the data node
- How many times the data files have been replicated
- When does the name node start?
- How many data nodes constitute the name node
- Capacity of the name node and space utilization

Data node comprises

- Data processing
- All the processing data that is stored on data nodes and deployed on each machine
- The actual storage for the file being processed
- Serving read and write requests for the clients

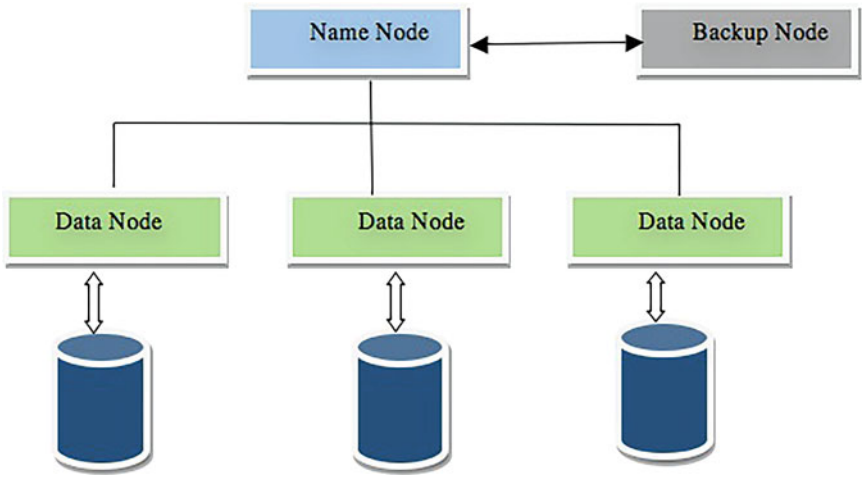


Fig. 20.1 Hadoop distributed file system

20.2.2 Map-Reduce

HDFS works with map/reduce to divide the data in parallel fashion on local or network nodes. The parallel structure requires that data is immutable and cannot be updated. It begins with input files where data is initially stored typically residing in HDFS. The input files are split up in input format, which selects the file, defines the inputs splits, breaks the file into tasks and provides the place for the record reader objects. The input format defines the list of tasks that makes up the map phase. The task then assigns to the node of the system based on where the input file chunks are physically resident. The input split describes the unit of work that comprises a single map task in a map-reduce program. The record reader loads that data and converts it into (key value) KV pairs that can be read by the mapper. The mapper performs the first phase of the map-reduce program given the key and the value, the mappers export key, and value pairs, and sends these values to the reducers. The process of moving map outputs to the reducers is known as shuffling [28] (Fig. 20.2).

The set of intermediate keys are automatically stored before they send to the reduce function. A reducer instance is created for each reduce task to create an output format. The output format governs the way objects are written, functioning similar to input format class as described earlier. The output format provided by Hadoop writes to the files to HDFS [27, 28].

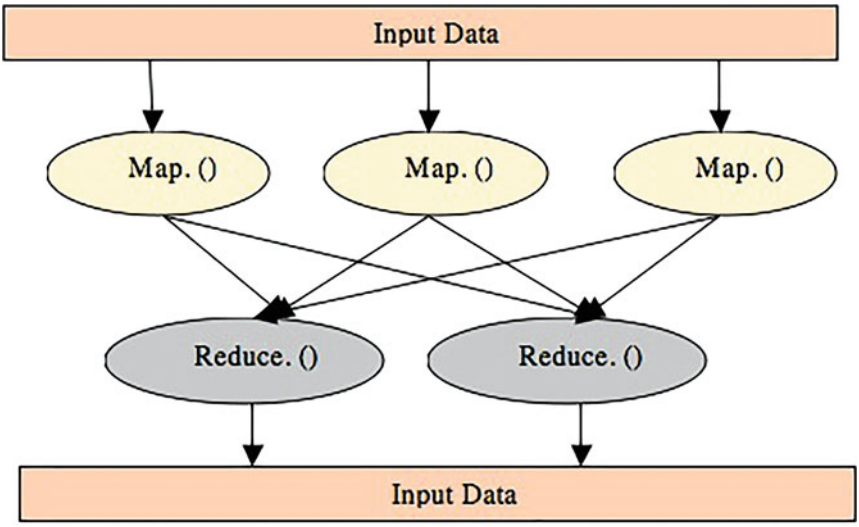


Fig. 20.2 Map-Reduce

20.3 Challenges and Opportunities with Big Data

Datasets grow in size over the years generated from enterprises, IoT, biomedical, and other sources. Imagine the amount of data being generated on daily basis from mobile devices, sensory technologies, software logs, cameras, and social websites. Data captured at high cost is often ignored and deleted due to lack of storage requirements but big data has changed the way we store and capture data [26]. The data generated by these sources are heterogeneous, which brings numerous challenges and diversity in solutions. Big data allows us to take educative decisions from structured or unstructured data as it contains great values, which cannot be explored in small sized data.

Retailers get to know their customer needs, attitudes, and motivations to buy. They use this information to create a competitive advantage by making relevant personalized offers to individual customers. Manufacturers are using big data to determine optimal maintenance cycles to replace component parts before they fail, thus increasing uptime and customer satisfaction. Pharmaceuticals are using big data to accelerate drug discovery and offer more personalized medicine. Government agencies are using big data to protect against cyber-attacks. While the reward for the successful big data initiative can be game changing, the reality is, there are some challenges to overcome.

To store and access huge volume of data, companies needed to go for either horizontal or vertical scaling. In horizontal scaling, multiple independent machines are added to distribute the workload across many servers with multiple operating systems running on each. Companies normally contact database vendors for a bigger solution, which requires significant investment. With horizontal scaling we can increase the performance in small steps and thus minimizing the upfront investment. One can scale out as much as needed but data distribution and parallel processing complexities needed to be handled at the software level. Vertical scaling involves a high-performance machine with more memory, storage and processing capabilities, and easy maintenance; high availability of software is added advantage in vertical scaling. But at the same time there is limit on the amount of scaling that can be done and also upfront cost to increase the hardware capabilities [26].

The solution for big data comes from the interdisciplinary activities and can be broadly categorized in management science, information science, match and statistics, and engineering. Management science involves in data acquisition and data management while information science includes data access and processing. Data understanding comes under mathematical activities and application development is the engineering activity [29].

The semi-structured and unstructured data needs to be transformed to structured data as the current data mining or machine learning algorithms are not optimized to handle huge chunks of unstructured data. The current technologies are also lacking the computing capability to handle the big volume of unstructured data [30]. Big data technologies have become commodities for mass adoption provided by Hadoop platform for storing and processing big data. Most of the data storage

solutions reside in clouds. Data collected from different resources is inconsistent and redundant which needs pre-processing for effective data analysis. The pre-processing may involve in providing a user with a uniform view of integrated data, which is collected from multiple resources. To improve the data quality and consistency data needs to be purified by identifying incomplete, un-reasonable and in-accurate data. Redundant data not only cause transmission expense but also use storage resources. This redundancy needs to be eliminated for data consistency and reliability. Although data redundancy methods help to remove redundant data but on expense of computation cost, e.g., data compression and decompression especially for video data which has temporal and spatial redundancy [29, 30].

The following sections provide a detailed survey of opportunities, challenges, and issues of big data (Hadoop) in terms of data locality, load balancing, heterogeneity issues, scheduling issues, in-memory computation, multiple query optimization, and I/O issues, respectively

20.4 Data Locality and Data Placement Issues in Hadoop

MapReduce is widely used for distributed parallel processing and data intensive applications. In Hadoop, the input data file is divided into different data blocks. These blocks are distributed into many nodes in the cluster. One of the important concepts of the Hadoop is to move the computation instead of moving the data when we are dealing with a large amount of data. It helps to improve the performance and network congestion of the system. The following section describes some of the research efforts to optimize the performance of Hadoop in terms of data locality.

Hadoop implements three levels of locality. First level locality is node locality. It is most efficient locality where it processes the map tasks. Second level locality is rack level locality; if a task is failed to achieve the first level locality then scheduler launches the task where computation node and data node are on the same rack. If still it cannot be able to get the second level locality, then it again starts the job on the node having the different racks; it is also called the rack off level. Straggler problem occurs due to the unavoidable runtime clash for the transmission capacity of system, processor, and memory. Radha et al. [31] proposed the speculative execution performance balance where data locality can fulfill by slot prescheduling. Moreover, delay scheduling is feasible for improving the data locality of MapReduce [2, 3]. They also proposed the dynamic map reduce, concentration of Hadoop fair scheduler as compared to FIFO scheduler. It has three main components, i.e., rescheduling of slots, speculative execution balance, and slot allocation of Hadoop dynamically. Primary objectives of proposed approach are to improve the job execution time on the cluster and enhance the performance of MapReduce. Data locality minimizes the network traffic and bottleneck in data intensive applications. Traditional high-performance computing (HPC) has separate computation and storage. For multiple users, it fulfills the needs by providing the high-speed link interconnection. But the capacity of these links is very less

compared to the bandwidth of a computing node. Zhenhua et al. [32] explored the data locality by providing mathematical model for the MapReduce scheduling and investigated the impact of data locality theoretically. They investigated the scheduling of Hadoop and proposed an algorithm. The main objective of algorithm was to schedule the multiple jobs concurrently instead of one by one for the optimal data locality.

Eltabakh et al. [33] introduced co-Hadoop which is an extension of the Hadoop framework. It enables the applications to control where data is stored. There is no need to convert the data into the certain format. Instead, there is an application that will help by giving the hints to co-Hadoop that some set of files are related and need joint processing. They developed this technique in such a way so that it can achieve the fault tolerance of Hadoop. They argued that colocation could be used for the improvement of efficiency of various operations having aggregation, columnar storage indexing and joins in a context of log processing. For the processing of the log they find out two use cases, one is sessionization and second is joins. The main aim is to speed up query processing and showed that if both co-partitioning and collocating are used together, higher performance can be achieved on sessionization and joins. MapReduce is very popular programming model for processing and analysis; it has limitations such as an application created for a single cluster cannot be work for large-scale data processing for distributed environment. It also has limitation for distributed data processing such as efficiency and reliability. Wang et al. [34] have introduced G-Hadoop, a framework of map reduce, whose main objective is to provide the large-scale distribution among the various clusters. The need for data intensive analysis among many distributed clusters for scientific data has risen significantly nowadays. To overcome these issues they designed a new architecture of G-Hadoop, which has master and slave computation model. They also stated that existing cluster could be added G-Hadoop with little modification.

There are many issues and challenges in the processing, storage, and analysis of big data. However framework like MapReduce can manage big data by processing and distributing data among the various nodes. MapReduce allows the users to perform the task without having the internal knowledge and detail of the system. But the problem occurs when it works in the heterogeneous environment. Hsu et al. [35] introduced new approach to improve the data locality and load balancing using by virtual machine mapping. Before the mapping, it dynamically divides the data and utilizes the resources of using the virtual machine mapping in the reduce phase. The primary objective of this technique is to enhance the performance of MapReduce in the heterogeneous environment. The advantage of using this strategy is to minimize the overhead in distributed system and to optimize the shuffling performance and workload balancing at runtime. The experiment shows that proposed technique enhances the performance of MapReduce in terms of data locality and total execution time. Performance of Hadoop map-reduce cluster can be improved by avoiding the off-switch communication. Presently, a primary focus of Hadoop is to minimize off-switch task of a map phase. Grouping of blocks in racks could significantly minimize the exchange of off-switch data and thus can decrease the execution time. Yu et al. [36] proposed this approach to reduce the off-switch

data access and execution time. They also find out the loss of parallelism due to sticky effects and conflicts during the grouping of blocks for the enhancement of the data access. To minimize these effects, they also introduced the task scheduling and data placement to reduce the impact of grouping blocks approach on parallelism. They also conducted the experiments to validate their approach and show that execution time of the job reduced up to 56% with minimum loss of parallelism.

MapReduce performs extensively join operation during the processing of a large amount of data and traditional approaches are not effective due to the partitioning skew and take long to response for the execution. Lin et al. [37] proposed the skew avoiding and locality aware algorithm by using volume aware locality in spite of hash partitioning. It distributes the data and reduces equally only when there is skew data and also transfers the data based on the locality of the network without any modification in MapReduce framework. They have also checked the performance of the proposed method in terms of effectiveness of partitioning, a degree of skew, and response time of the join operation. Rhine and Bhuvan [38] proposed locality aware scheduling and splitting strategies to improve the performance of MapReduce. In locality aware scheduling approach, it checks that slot is available for the local data. On the other hand, input-splitting method verifies that the cluster data blocks from the same node splits into single partition. They executed splitting and scheduling methods separately, to show the better performance without any modification.

Data intensive applications and cloud environment bring new challenges for the data such as computation, assessment, and placement. Network traffic plays a major role in the performance of data intensive applications. Some of these existing challenges can be reduced by increasing the data locality in the distributed applications. Chen et al. [39] addresses the data locality issues in Hadoop and proposed the locality-aware scheduling (LaSA) algorithm. In large-scale Hadoop clusters improving the data locality of the MapReduce is very important by moving the computation instead of data. Optimal tasks scheduling can minimize the overhead of network traffic, which is essential for the efficiency and stability of the system. Most of the available schedulers do not consider the data locality during the reduce phase which affects the performance of the application. To improve the data locality, Tan et al. [40] applied the threshold-based optimal placement for the reduce task to reduce the fetching cost of the data. The issue of a partitioning skew due to a massive amount of data transfer during the shuffling phase is one of the major bottlenecks which causes a significant impact on the performance. Skew partition causes the network congestions due to the blind partitioning. Ibrahim et al. [41] proposed a novel locality and fairness aware key partitioning (LEEN) methodology. They argue that locality and fairness are important factors for the data partitioning in MapReduce. The proposed technique solved this issue by providing the asynchronous scheme for the map and reduce phase. It has full control on the key distribution during the reduce phase and improves the data locality and fairness of MapReduce. Experiments results show that proposed technique improves the overall performance of the Hadoop by 40% compared to the default technique.

To improve the data locality and load balancing, Panchputre et al. [42] proposed various steps. Affinity is calculated using the mathematical formula, then they assign

the weight to the various components based on the location of the server of a region. They also evaluated the proposed scheme and found out that locality aware balancer performs better than the simple load balancer. Wang et al. [43] have described that load balancing is essential for the disturbed system and task to achieve the best performance. They argue that in work stealing environment, tasks are randomly transferred from one heavy loaded scheduler to the idle scheduler. However, the data locality of the data intensive applications where the larger amount of data is transferred blindly is an important factor. To solve this issue, they tried to improve the work stealing approach by organizing that task queue on the basis of task location and its size. They implemented this approach using the MATRIX that is a platform for the multiple tasks computing scheduler. Using this method, they tried to achieve load balancing and data locality. The distributed key value pair is used to organize and manage the metadata, of tasks. They evaluated their approach by using the different scheduling policies for various applications.

Park et al. [44] proposed dynamic resource reconfiguration (DRR) scheme for data intensive applications to maximize the resource utilization of the individual virtual machine (VM) by increasing its capability. Using this technique, they tried to improve the data locality of the overall system. Hadoop and MapReduce framework have scheduling issues which are addressed by Zhang et al. [45]. Over the years data locality is widely investigated for the cloud environment. They proposed data locality aware scheduling for improving the performance of the MapReduce framework in a heterogeneous environment. It receives the request from the node and schedules the task preferentially on the requesting node; if it does not exist then will go for the near- by node and decide whether it schedule the task or reverse the task for the storage of data for input by sending the data on the fly. For the evolution of the technique, they compared the proposed method with default Hadoop scheduling and find out it improves the data locality and decreases the execution and response time.

It is very costly to fetch data from the remote server in the large processing of data. So, it is necessary that data should be co-located with computation. Fan et al. [46] proposed a dependency aware data locality for the map reduce (DALM). It is a replication-based approach for the general input real-world data that is dependent and skewed. DALM adjusts the dependency of the data in locality framework and has two main parts, one is computing factor of each file and second is replica placement. The proposed methodology is compatible with traditional infrastructure and can be extended to the other distributed paradigm of computation. Khan et al. [47] investigated the various scheduling algorithms, i.e., DARE scheduling, delay scheduling, matchmaking scheduling, prefetching, pre-shuffling, and next-k-node algorithms. The basic purpose of these algorithms is to improve the data locality. They also evaluated these algorithms and found that these algorithms solve the data locality problem but raise many other issues like cost of replication and an overhead of computation.

20.5 Load Balancing

Load balancing can be broadly categorized as static and dynamic. Static load balancing techniques are usually based on the greedy algorithms, search algorithm, and machine learning algorithms. Usually, overheads occur at the start of the static load balancing. In dynamic load balancing, there are also various techniques such as stream based, cloud computing, and discrete event simulation. Usually, map and reduce stages are used to parallelize and process large amount of the data efficiently but data skew problem causes data imbalance. Ye Chen et al. [48] proposed a new partitioning cluster locality algorithm (CLP). It consists of three parts. The first part is pre-processing; in this part, they select the sample from the main collection of data, so that they can understand the distribution of data. The second part is clustering; in this part data is formed from the various cluster data. Data that have the same key kept with the same cluster so that it can reduce the size of data. Third and final part is partition of locality; it assigns the data to the cluster suitable according to data locality. They also analyzed according to the execution time, data locality, and skewed degree. They showed that proposed technique is better than default technique but comes with large overhead due to extra Mapreduce phases.

Hadoop applications run as containers, so the problem like concurrency affects the job completion time and resource usage of the system. When there are too many concurrent containers, resources bottleneck occurs, and system resources are underutilized. Kamal et al. [49] introduced a new approach, concurrent container slot (CCS) and tried to enhance the performance of Hadoop applications. The proposed method is dynamic, and uses the controllers that take instantaneous score or score to CSS ratio as input and generate the new CSS as output. This score is a combination of user CPU, blocked processes, and context switches values. They also assessed the water level, PD, and PD + pruning controller. On the other hand, a dynamic controller has the better performance. To make sure that their work applies to the all of the map reduce applications, they selected the six applications in this work to have diverse usage of IO and CPU usage profile. Finally, they find out that using the dynamic tuning offer better performance instead of using the existing default best settings. Indranil et al. [50] proposed parallel-boosting algorithm and also define a framework for the MapReduce to improve its performance. They analyzed the performance of the proposed algorithm by using the Amazon EC2 to show improved speedup, accuracy, and scalability. Sui et al. [51] have surveyed various techniques and frameworks for the load balancing and data intensive computation. They also explored the various data intensive frameworks such as MapReduce, Hadoop, and Dryad framework.

Web-based applications have a large amount of data and handing the millions of users is one of the challenging tasks. Traditional methods to process the data are inefficient due to computation cost. Most of the frameworks proposed over the years to handle these issues are primarily focusing on homogeneous data, but a performance of these frameworks suffers while dealing with heterogeneous data.

Ajitha et al. [52] introduced a dynamic allocation framework, to allocate available resources based on the direct acyclic graph. The proposed architecture consists of job manager, which is responsible for managing the jobs, and divides the jobs into tasks. The second component is task manager, run by the virtual machines, receives the new task, executes it and informs others about its completions. The third component is a load balancer, it uses the join idle queue algorithm, and whenever task manager becomes idle it joins the queue. The main objective of proposed approach is to enhance the throughput, load balancing and decrease the latency in the heterogeneous environment. Data is not co-located in Hadoop by default, which affects its performance and can be resolved by grouping and partitioning the log processing operations, i.e., grouping joins, sessionization, and indexing. Nishanth et al. [53] proposed a new approach that achieves the load balancing and colocation of HDFS data blocks. It was developed on the co-Hadoop. It divides the input files by grouping that is an attribute of the similar data. The same set of data is co-located so that similar data can be partitioned. So the problem of load balancing can occur due to the skew data. So they enhanced the co-Hadoop and introduced the new algorithm to solve the problem of load balancing. Proposed scheme ensures the load balancing, fault tolerance and reduces the application execution time.

Data skew occurs in various applications like mining of data, operations of joins, and graph-based applications, resulted in load imbalance. Some approaches handle the skew of the data, but it leads to additional overhead and computation cost. Xu et al. [54] proposed a new technique that has two phases, one is sampling and second phase is job execution of MapReduce. In sampling phase, it computes the key frequency approximated distribution to make the good partitioning scheme. In MapReduce job execution phase, it applies the partitioning scheme on each step for grouping the keys quickly. To achieve the load balancing, they further proposed the cluster split and cluster combination methodologies. The proposed technique reduces the overall execution time by improving data locality. Data skew causes some tasks to take longer execution time and degrade the overall performance. Chen et al. [55] proposed the lightweight data skew mitigation technique to solve the load imbalance problem. As compared to the existing approaches, it does not need sampling at runtime. It is an innovative method to balance the load among the reduce task that encourages to split the large keys when there is the semantics of applications. It also modifies load of MapReduce in a heterogeneous environment. To attain the goal of load balancing, they also try to achieve the transparency, parallelism, accuracy, large cluster splitting and tackled heterogeneity issue to enhance performance. Zhou and Wen [56] have proposed the novel scheme to improve the load balancing in Hadoop. The proposed scheme is based on the user history for specific access of data and mixture of analytical process hierarchy (APH) for the load balancing. It checks the history of file access, a capacity of the machine, CPU and memory utilization.

MapReduce has good performance in a homogenous environment, but in the heterogeneous environment its performance is suffered and takes a long time to execute the job. When various nodes work together on the same amount of data, problem of load balancing arises in heterogeneous environment. Gao et al. [57] proposed a new scheme for load balancing based on the performance of the node (LBNP). It is based on the history of the evaluated performance of the node and assigns the task according to the performance of each node. They tried to enhance the efficiency of the MapReduce by shortening the tasks of the reduce phase. There are also some limitations in the existing load balancing techniques, i.e., static load balancing is less effective in a large size cluster. Fadika et al. [58] proposed a MapReduce with adaptive load balancing for heterogeneous and load imbalanced cluster (MRLA). The proposed technique is suitable for both homogeneous and heterogeneous environment. They also compared their technique with existing techniques, and enhanced performance is reported. Shi et al. [59] addresses the relocation of the shuffling phase in MapReduce and aimed to reduce the network traffic, balance the workload, and remove the network hotspot for improving the overall performance. To achieve these goals they have introduced a new technique and name it as smart shuffling scheduler. They showed that proposed scheduler outperforms CoGRS and random scheduler. Myung et al. [60] proposed a multi-dimensional range partitioning (MDRP) to address the data locality and skewness issues. The proposed technique solves the limitations in two ways, one to handle the data skew in the better way and second, to handle the degree of skew data. They also performed experiments with the proposed technique and compared it with the range-based method. The proposed method is 6.76 times faster compared to its counterpart without any modification to original MapReduce.

20.6 Heterogeneity Issues

As Hadoop implementation assumes, most of the map tasks are data local and computing nodes are homogeneous, which are not always true, as in virtualized data centers. Map-reduce performance is severely affected if data locality is not taken into consideration in heterogeneous environment. Xie et al. [61] introduced a technique for placing data across the node in a way that each node has a balanced data for the processing. They also identified performance problem in HDFS (Hadoop Distributed File System) on heterogeneous clusters. Motivated by the performance degradation caused by heterogeneity, they have designed and implemented a data placement mechanism in HDFS. The proposed scheme distributes fragments of an input file to heterogeneous nodes based on the computing capabilities. Rajashekhar et al. [62] proposed a mathematical model; based on the specification of hardware, it estimates the computation ratio. The model calculates the computational ratio by resources available at execution time. MapReduce consists of various nodes that are

different according to the computing capacity of the different nodes in cluster. They assumed that it is necessary for data placement algorithm to divide the input data based on the computing capability of a node in clusters. They proposed a method, which is based on the history that calculates the computation ratio of a machine when a job is completed.

As Hadoop assumes that each node in a cluster has the same capability of processing, the data and task are local. But on the other hand, it decreases the performance and increases the overhead MapReduce. To overcome these issues, Lee et al. [63] proposed the dynamic data placement (DDP) policy for mapping task of data locality to allocate data blocks. DDP consists of two different phases, in first phase input data is written into the HDFS, in the second phase given job is processed. By default, Hadoop policy of data placement can be applied to the homogeneous environment, but in a heterogeneous environment, it creates problems like load imbalance and unnecessary overhead. They introduce the DDP algorithm to solve the problem like load balancing, data locality, and overhead reduction. For the analysis, they did experiments on the two types of job, word count and grep, to evaluate the performance of proposed scheme in Hadoop heterogeneous cluster. They find out that DDP can improve the word count up to 24.7% with an average of 14.5%. But in grep, the DDP improves up to 32% with average enhancement up to 23.5%. Hadoop is the well-known framework to deal with extensive data of distributed applications among the many clusters of commodity servers. A main advantage of the Hadoop is that it handles failures. Hadoop has by default homogeneous scheduling approach for the processing of various jobs. But in the cluster, performance of Hadoop suffers due to a heterogeneous environment. To overcome this issue, Sujitha et al. [64] proposed the new technique for heterogeneity and data locality for Hadoop. In this approach, they define a new architecture, which consists of various levels. The first level includes an operating system, which controls the hardware; in second tier there is Java virtual machine. The third level consists of three components: (1) processing of data, (2) data storage, (3) coordination; the core purpose of these components was to make processing and storage, fault tolerant, fast, and scalable. Fourth and fifth level consists of network and application layer, respectively. They also evaluated their technique in terms of response and execution time, fairness, velocity, and mean time for completion.

Hadoop supports homogenous task and due to the heterogeneity, Hadoop faces significant challenges in cloud environment. Existing techniques have several issues, i.e., consuming unnecessary bandwidth, not useful when history is not accessible, and cluster size increasing the complexity of the system. To address these issues, Ubarhande et al. [65] have proposed the new data distribution technique for the heterogeneous environment. Distribution of data includes the calculation of capability of processing of every node specified in Hadoop cluster. Algorithm begins with response of the log file of each slave node, which is generated by the speed execution analyzer. Using the proposed technique, they tried to improve the response

time of the Hadoop system. Moreover, they also verify the proposed technique whether the response time is increased or not; they tested their technique using the two applications of MapReduce. They find out that the performance of the Hadoop is improved in a heterogeneous environment when data size is increased. Hadoop MapReduce allows user to flexible customization and the convenient usage. There are some issues in the processing of a large amount of data; one of the issues is straggler task, which delays the processing of the job due to the slow running tasks. Huang et al. [66] have proposed two techniques, one is estimated remaining time using linear relationship model (ERUL) and second is extensional maximum cost performance (ex MCP). It is dynamic load aware approach and also performs well and overcomes the issues of the state-of-the-art approach such as LATE and ex MCP. In MCP, they also tried to allocate the slot's value that is ignored in the traditional MCP. The experiment results show that proposed approach estimates remaining time of the task efficiently and also detects the stragglers accurately.

20.7 Scheduling Issues

There are many challenges existing in Hadoop, which ranges from the job scheduling, locality of data processing, usage of resources efficiently, and fault tolerance. Hadoop uses by default first in first out (FIFO) scheduler and uses the two different types of scheduling policies, one is job level and second is task level. Therefore, task assignment is a responsibility of scheduler; a job will be moved to the queue when submitting this job. From the job queue, the job is divided into tasks and distributed into different nodes. By using the proper assignment of tasks, time of the job completion can be reduced. The performance of Hadoop framework depends on the hardware configuration of each node and cluster size. Prasad et al. [67] analyzed the performance of scheduler for multi-jobs in the Hadoop cluster, i.e., homogeneous and heterogeneous workload. They have compared different schedulers, which are default scheduler (FIFO), fair scheduler, and capacity scheduler. They compared these schedulers regarding job execution time and waiting time for a job. They find out that default scheduler takes less time when a job is small but takes the more time when the job is bigger. The fair scheduler takes the less time for job execution, and the capacity scheduler consumes more time for the job execution as compared to the fair scheduler. Job scheduling in MapReduce is an important and critical issue that affects the performance of Hadoop framework. Sethi et al. [68] introduced a new algorithm that runs the high priority jobs on the free nodes. Scheduler will assign tasks on some other nodes based on the availability. The primary objective of the proposed technique is to process the jobs locally to improve the throughput and response time. Moreover, they identify the problem with delay scheduling regarding the overhead on job tracker. The proposed algorithm reduces the cost on the job tracker by distributing the load on task tracker.

Zaharia et al. [69] have proposed a new technique for the cluster scheduling to achieve the locality and fairness. There exist a tradeoff between fairness and locality as performance of both is interconnected. So to solve the fair sharing problem there are two approaches, one is to kill the task or wait for the job to be completed and then submit the new job. The second approach is to achieve the data locality. To address these issues, an algorithm is proposed called delay scheduling that addresses the fairness and data locality issue. They also argued that delay scheduling is not performing well for the environment like Hadoop. But this approach is not suitable where the fraction of a task is longer than a job and there are few nodes per slots. However, they believe that it can improve by two things first by short tasks and usage of multicore environment. They also discussed various ways for the generalization of delay scheduling, i.e., scheduling preferences, load management mechanism, scheduling policies, and distributed scheduling. They also implemented the delay scheduling in Hadoop fair scheduler (HFS) which improves the throughput for a heavy workload and also improves the execution time for shorter jobs. Sun et al. [70] proposed the high-performance scheduling optimizer (HPSO). The main objective of HPSO is to decrease the execution time of MapReduce and improve its overall performance. HPSO consists of three modules prefetching, scheduling optimizer, and prediction. The main job of scheduling optimizer is to predict the suitable nodes of task tracker in which upcoming task will be allocated. Once the decision of map task scheduling is done, than HPSO loads the estimated data using the prefetching module. HPSO improves the data locality for the MapReduce by prefetching. HPSO works by predicting the most suitable node for an already pending task by determining when to prefetch the task. For launching new task, it preloads the data required to the memory without delay. It also minimizes the time of map task and decreases the time of MapReduce job.

Gu et al. [71] introduced the optimized Hadoop named as SHadoop to improve the performance of MapReduce. It has two main phases, job initialization and job termination. It provides the instant messages for the communication for efficient and important event notifications instead of heartbeat mechanism. The proposed approach is evaluated with various benchmarks and approximately 25% improvement is reported compared to default approach. Yang et al. [72] introduced a new scheduling algorithm that takes Map and Reduce phases as two separate phases in a homogenous environment. They have proposed a novel one to one sampling technique that calculates the average time of execution for the MapReduce tasks. They also argue that proposed methods take fewer resources for the map and reduce phases to determine the characteristic of job and which job property is more suitable. The proposed algorithm improves the efficiency and real-time scheduling for the MapReduce. Sadasivam et al. [73] addresses the problem of task assignment in the heterogeneous and homogeneous environment by applying the hybrid particle swarm optimization genetic algorithm (HPSO-GA). The operation of GA such as mutation and crossover utilizes the resource of data intensive application efficiently and completes the task within given time. The main objective of proposed algorithm

is to divide the workload of each node according to the processing capacity of a node in the heterogeneous environment. The proposed algorithm is a combination of both particle swarm optimization and genetic algorithm. The operations of particle swarm optimization (PSO) and genetic algorithm (GA) are applied in specific particles. Allocations of the tasks are based on the fitness value, to balance the load. There are various steps in this algorithm; initially, it initializes the count of the generation, size of the population, and maximum generation. It generates the initial population of particles randomly. It calculates the fitness value of the population and updates the generation of population and velocity after they perform the crossover operation. They also evaluate the fitness function of the particles. They continue this process until they get the global best particles. HPSO-GA applied on the MapReduce framework improves the utilization of the resource and load balance in a grid environment. Reliability, efficiency, and scalability of MapReduce also improved with the proposed methodology.

Li et al. [74] analyzed the LATE scheduling algorithm to analyze how much time needed for a particular task to complete. The main drawback of a LATE algorithm is its inability to deal with data locality in cloud computation. It executes reduce task without considering data is local or not. On the other hand, it executes the map task locally. To improve this algorithm, they have stated following steps. One checks the speed of the node; if it is slower than the threshold, it is better to ignore it; otherwise continue the processing. According to the request of task node in the particular rack, make sure that if task is slow then find out how much time is remaining to complete that task. The third step is to make sure that if task being executed in other rack is slow then keeps it to another queue. The fourth step is to find out whether the task has an access to data locally, if it doesn't have, than make sure the remaining time of a task is longer than a threshold value. Their experiments show that the proposed enhancement reduces the response time and improves the throughput of the system.

20.8 In-Memory Computation

Zaharia et al. [75] proposed a resilient distributed dataset (RDD), so that developer can able to perform the in-memory computation on the massive cluster in the fault tolerant way. They addressed two major issues, inefficiency in case of an iterative algorithm and interactive data mining. In both case data is not stored in the memory for computation instead the data is stored on hard disk, which needs multiple accesses and becomes the bottleneck for these algorithms. Moreover, existing frameworks reuse the data by storing it to the stable external storage. It also causes an extra overhead for the disk I/O, replication of data, and serialization, resulting in increased execution time of an application. The proposed technique RDD has a parallel data structure and optimized data placement, and it provides the fault tolerance, having the rich manipulation operators. Cliff et al. [76] have

presented the coarse-grained distribution memory for data analysis which allows user interactive query processing and deep analytics for large dataset. They also argue that it is essential to manipulate in-memory computation for the sophisticated data analysis, which is valid for the machine learning algorithms as these algorithms are iterative in nature and have a strong temporal locality. The traditional database uses the fine-grained technique, which works only on the single record. In case of a large number of nodes, fine-grained approach is hard to scale and provides the failure recovery. Matei et al. [77] proposed a framework called spark that enables the fault tolerance and scalability of map reduce for in-memory computation. They have introduced the resilient distributed dataset (RDD) (read only object collection) and are used to rebuild the portion, which is lost. Spark also consists of two types of variables; one is broadcast variables and second is an accumulator, which makes more reliable processing on the large dataset.

20.9 Multiple Query Optimization

The most common problem that users are facing is to query data over Hadoop. The user uses the traditional infrastructure of RDBMS and is not aware of an internal aspect of MapReduce, for data extraction from the warehouse. Hadoop Hive is an open source tool to address these issues. Dokeroglu et al. [78] reviewed the multiple query optimization (MQO), which gives higher performance over decision support system [79, 80]. They proposed the shared hive to improve the performance of Hadoop hive. The proposed approach processes the query of HiveQL for improving the execution time. In shared hive, they have detected the common task of TPC-H and Hive Query and merge them together to form the global set of insert query. Dokeroglu et al. [81] have analyzed the performance of MapReduce based query execution on the Hadoop using the different virtual resource settings and observed the performance results under each setting. The distributed environment of Hadoop takes advantage of the more power of CPU and RAM capacity than the network bandwidth, for higher performance. On the other hand, data warehouse with larger sizes consumes the greater network bandwidth as compared to the smaller data warehouses.

Spark provides the feature like fault tolerance and allows users to run the query with much faster rate, and machine learning code speeds up as compared to the Hadoop. Unlike a traditional system, spark shows that it is possible to achieve the speedup while keeping the execution engine of MapReduce and without compromising the fault tolerance. Xin et al. [82] have extended the execution engine of spark by providing the in-memory computation based on column orientation and re-planning the mid query for efficient execution of SQL. Spark uses both MapReduce and traditional techniques into column-oriented storage and partial direct acyclic graph (DAG) implementation.

Guoping et al. [83] presented two different techniques to optimize the multi-query on the MapReduce. A first technique is multi-job optimization for the MapReduce framework as it merges the multiple jobs into a single job, so it can scan and share the input file and output. They have also proposed multi-query optimization based on the materialization that supports multi-jobs sharing. An algorithm is proposed to organize jobs in optimal way with optimal processing to each group. Cherif et al. [84] proposed new technique on the basis of HadoopDB. The primary objective of this technique is to improve the performance of Hadoop by adding more components to the system. They also introduced a new space efficient data placement technique for the MapReduce warehouse and SQL-translator. Higher performance, flexible query interface, scalability, and fault tolerance are main concerns for optimal and efficient query processing. The processing of a large amount of data and efficient analysis on the cloud is critical. The essential thing for processing and analysis of data is the similarity of joins. Silva et al. [85] proposed multi-round MapReduce similarity joins (MRsimJoins). A multi-round MapReduce based algorithm efficiently solves the problem of similarity of joins. It divides the data and distributes among the node until subset is small enough to process by a particular single node. The process is also divided into various rounds. An input data is partitioned by initial round and previously generated partition done by the subsequent round. MRsimJoins executes the data until all data is processed. Two types of partitioning are used for the data using K pivots, one is window pair and other is base partitioning. Window pair partitioning has all the recodes in boundary between two base partitions. Base partitioning consists of all records that are close to the pivot than any other pivot. So the proposed technique is more general and it can be applied to any dataset.

Dan Suciu et al. [86] have presented an algorithm for the evaluation of query and edge label graph on the semi-structured data. This algorithm also provides the efficient communication steps and transferring of data during the assessments. However, they mentioned that algorithm has an issue like communication data is gathered from the single site. It leads us to the bottleneck, when we evaluate it for the real-time data. There is also another problem that the performance is also suffered when links are quickly increased in social graphs. To overcome this issue, they proposed two enhancements in the existing algorithm. The first enhancement is one-pass algorithms to remove a large amount of data. Second, the iter_acc algorithm using the MapReduce programming model in an environment of Hadoop. The fundamental purpose of this is to solve a performance-suffering problem. They also performed experiments, which shows that proposed one-pass algorithm removes the redundant data 50% during the evolution process and second algorithm performs well even if the input data size is doubled. Nowadays there are many cost effective solutions provided for the cloud platform, for the effective and efficient query processing. But currently, many researchers focus on optimizing query performance without considering the characteristics of query. So without bearing in mind the similarity of query, it can cause the redundant data computation that also affects the efficiency of query. Ding et al. [87] proposed a framework based on the multi-query optimization called the multi-Q. The multi-query divides the search range

of query into the set of query reuse units (QRU). It also manages the similarity among the incoming queries by managing the multi-query reuse dependent graph (MQDG). Based on the MQDG, they showed how to process the query in multi-Q. They manipulate the cluster-based partition algorithm to conduct the logic partitioning of the query. They implemented the multi-Q based on the SEU-Cloud and they did experiments for the demonstration of efficiency and effectiveness of their approach. They also compared their technique with Hive with approximately 39% of improvement.

There are various challenges in the traditional system for processing of the query on huge volume of data. The core issue lies in the traditional indexing structure such as R-Tree and B-Tree are naturally centralized and crippled when it is employed in a distributed environment. Aly et al. [88] built the system called the kangaroo to improve the execution of queries over the range of data. It also divides the data into the various non-overlapping partitions to reduce the query execution time. Workload awareness, no duplication of data, no partition overlapping and size are some of the characteristics on which the proposed technique is based. It also has four primary processes through which a query is processed, i.e., loading the data, data partition planning, data partitioning, and query execution. They evaluated the effectiveness of their approach by checking the selective query effects, i.e., effects of search space scale and data skew on the partition size.

20.10 Input/Output Issues

High-End Computing (HEC) machines are creating the massive amount of data on single execution by running the big data application. The huge volume of data increases the complexity of scientific discovery, pattern analysis, decision making etc. The ability to move, store, analyze, and visualize huge volume of data is also a challenging task. The vast amount of growing data also imposes various requirements on the I/O performance; it becomes a reason of bottleneck in modern HEC machines. Zou et al. [89] presented the data analytics, how to minimize the data movement and release the severe I/O performance congestion. For a solution of this problem, they also build a quantity model for the evaluation of analytics algorithm and placement methodology. They presented a flexible analytics framework for the I/O performance optimization. It is flexible placements approach, which combines the both visualization query and data compression. It consists of two algorithms, which dynamically choose and change to attain the best I/O performance with features describing and real-time resources observing. They also analyzed the bandwidth analysis from simulation to storage, analysis of bandwidth from simulation to staging nodes, analysis of latency with data query, and analysis of throughput with data query. Their results show that two important factors that have the significant impact on the analytics selection are data reduction ration and availability of processors.

Li et al. [90] have proposed a new architecture for Tachyon that is capable of read and write by preventing the synchronous data replication on write at the speed of memory. They also described that more data in memory becomes the bottleneck for any application. To overcome this issue, they also introduced recovery scheme based on lineage. By using the caching mechanism, read throughput could be improved for the distributed systems. Authors identified various characteristics of Tachyon e.g. immutable data, deterministic computation, and higher performance. The MapReduce needed the shuffling for a global exchange of the data that is produced by the mapping phase. The movement of the data across the disk causes a problem of I/O compound and contention. Yu et al. [91] introduces a new approach, which is virtual shuffling for the efficient movement of the data and reduction of the I/O for shuffling of MapReduce. The virtual shuffling technique is the combination of three techniques: three level segment table, balanced and dynamic merging sub-tree, and near demand merging. Three level segment tables manage the MapReduce tasks. It consists of a directory of segment table, a middle directory of segment table, and a global directory of the table. The data structure linked with these directories listed in the table. In balanced and dynamic merging sub-tree, they organize the entire virtual segment into a tree having leaves SEBs. In near demand merging, it waits until clarifying which particular segment requires the reduce function of reduce task. It does not wait for the last movement of the data fetch. As a result, they showed that proposed technique speed up the data placement, reduce the power consumption and solve the contention of I/O disk problem. Hadoop distributed file system (HDFS) is an open source and an attractive system for the processing of large dataset including parallel program processing based on the MPI, graph processing, and Java/Scala based framework to enable the user to do the iterative and interactive in-memory data analysis.

Yin et al. [92] investigated the problems in a parallel system as they find out that due to lack of intention in data distribution and access in HDFS, request for the parallel data sometimes serves as the imbalance and remote fashion. Due to this problem, it becomes I/O bottleneck for the storage nodes. To solve this issue, a new technique was introduced in which I/O middleware and match based algorithm are used for the optimization of data access in distributed file systems. To attain this, they get the data from the distributed storage and also perform the mapping relation among the process and chunk file where these files are associated with data processing task and operators. They also showed the relation in the form of bitrate matching graph. With this technique parallel data access used in such manner that it will be useful for the balanced and locality aware access to attain the higher I/O performance. At last, they have presented the content aware method for fast access to data without scanning the useless data from the large dataset. Table 20.1 presents the taxonomy of the related research efforts for each of the above-mentioned challenges.

Table 20.1 Taxonomy: challenges and issues of big data

Major	Minor	Primary	Secondary
Data locality	Data placement	Eltabakh et al. [33], Yu and Hong [36], Tan et al. [40], Wang and Chan [83]	DGM and graph based [85], scheduling [24, 28]
	Graph based	Xue et al. [94]	Scheduling [36, 40]
	Scheduling	Guo et al. [32], Chen et al. [26], Wang et al. [34], Park et al. [44], Zhang et al. [45]	Resource allocation [26, 44]
	Prefetching	Radha and Rao [31], Khan et al. [47]	Prescheduling [31]
	Virtual mapping	Hsu et al. [35]	Dynamic partitioning [35]
	Partitioning scheme	Lin et al. [37], Rhine and Bhuvan [38], Ibrahim et al. [41]	Join partitioning [37], input splitting/scheduling [38], key partitioning/fairness [41]
	Affinity/resource allocation	Panchputre et al. [42]	
	Replication based	Fan et al. [46]	
Load balancing	Partitioning scheme	Chen et al. [39], Nishanth et al. [53], Fadika et al. [58], Myung et al. [60]	Random sampling [39], colocation [53], cluster split [58]
	Resource allocations	Ajitha and Ramesh [52], Zhou and Wen [56]	Graph based [52], APH [56]
	Sampling	Xu et al. [54], Chen et al. [48]	Partitioning scheme [54], cluster split [48]
	Data skew	Gao et al. [57]	Node performance [57]
	Shuffling	Li et al. [90]	Scheduler [90]
	Parallel processing	Kc and Freeh [49]	CCS [37], parallel boosting [49]
Scheduling	Delay scheduling (fairness, data locality)	Zaharia et al. [69]	Fairness, data locality [69]
	Data locality (priority-based scheduling)	Sethi and Ramesh [68], Gu et al. [71], Sadasivam and Selvaraj [73], Li et al. [74]	Priority-based scheduling [68]
	Data placement	Sun et al. [70]	Prefetching based on node predictions [70]
	Resource aware allocation	Yang et al. [72]	Yang et al. [72] sampling
Heterogeneity	Data placement	Xie et al. [61], Arasanal and Rumani [62], Lee et al. [63], Ubarhande et al. [65]	Data partitioning [32, 33], load distribution [65]
	Speculative execution	Huang et al. [66], Sujitha and Jaganathan [64]	Heuristic approach [40]
	Load balancing	Arasanal and Rumani [62], Lee et al. [63]	Data allocation [34]
	Data locality	Sujitha and Jaganathan [64]	Scheduling [64]

(continued)

Table 20.1 (continued)

Major	Minor	Primary	Secondary
In-memory computation	Resilient Distributed Dataset (RDD)	Zaharia et al. [75, 77]	Optimize data placement [75, 77] RDD variables
	Coarse-grained	Engle et al. [76]	Scalability [76]
Multi-query processing	Virtual resources	Dokeroglu et al. [81]	
	Share scan task	Dokeroglu et al. [78], Wang and Chan [83]	Computational task [78], optimal processing [83]
	Fault tolerant	Xin et al. [82]	RDD [82]
	Similarity of joins	Silva and Reed [85]	Partitioning scheme [85]
	Data placement	Wang et al. [93]	Optimal processing [93]
	Graph based	Tung et al. [96], Ding et al. [87]	Distributed graph based [95, 96], partitioning scheme [87]
	Partitioning scheme	Ding et al. [87], Aly et al. [88]	Load aware [75]
I/O improvement	Asynchronous data replication	Li et al. [90]	Line age fault tolerance [90]
	Data placement/data preparation	Zou et al. [89]	Data placement/data preparation [89]
	Virtual shuffling	Yu et al. [91]	I/O contention [91]
	Optimized parallel data access	Yin and Wang [92]	I/O contention [92]

20.11 Conclusion

There are numerous challenges faced by the big data community when exploring huge volume of data to extract knowledge and meaningful information. These challenges include data capturing, transportation, storage, data analysis, management, visualization, etc. In this chapter, we focused on some of the core issues of big data by providing a detailed survey of opportunities and challenges of big data (Hadoop) in terms of data locality, load balancing, heterogeneity, scheduling, in-memory computation, multiple query optimization, and I/O issues. The taxonomy of the research efforts conducted over the years to overcome these challenges is also presented. The outburst of data generated over the last few years demands innovation in current processing technologies to process huge volume of unstructured data. The current technologies are also lacking the computing capability to tackle huge chunks of unstructured datasets. Real-time data analytics demands high-performance computing resources. These challenges demand solutions from multidisciplinary domains and need collaborative efforts to address these issues.

Acknowledgments The authors acknowledge with thanks the technical and financial support from the Deanship of Scientific Research (DSR) at the King Abdul-Aziz University (KAU), Jeddah, Saudi Arabia, under the grant number G-673-793-38. The work carried out in this chapter is supported by the HPC Center at the King Abdul-Aziz University.

References

1. Usman, S., Mehmood, R., Katib, I.: Big data and HPC convergence: the cutting edge and outlook. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 11–26. Springer, Cham (2018)
2. Mehmood, R., Faisal, M.A., Altowaijri, S.: Future networked healthcare systems: a review and case study. In: Boucadair, M., Jacquenet, C. (eds.) Handbook of Research on Redesigning the Future of Internet Architectures, pp. 531–558. IGI Global, Hershey (2015)
3. Alam, F., Mehmood, R., Katib, I., Albogami, N.N., Albeshri, A.: Data fusion and IoT for smart ubiquitous environments: a survey. IEEE Access. **5**, 9533–9554 (2017)
4. Muhammed, T., Mehmood, R., Albeshri, A., Katib, I.: UbeHealth: a personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities. IEEE Access. **6**, 32258 (2018)
5. Suma, S., Mehmood, R., Albugami, N., Katib, I., Albeshri, A.: Enabling next generation logistics and planning for smarter societies. Procedia Comput. Sci. **109**, 1122–1127 (2017)
6. Suma, S., Mehmood, R., Albeshri, A.: Automatic event detection in smart cities using big data analytics. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 111–122. Springer, Cham (2018)
7. Mehmood, R., Alam, F., Albogami, N.N., Katib, I., Albeshri, A., Altowaijri, S.M.: UTiLearn: a personalised ubiquitous teaching and learning system for smart societies. IEEE Access. **5**, 2615–2635 (2017)
8. Mehmood, R., Graham, G.: Big data logistics: a health-care transport capacity sharing model. Procedia Comput. Sci. **64**, 1107–1114 (2015)
9. Ahmed, W., Khan, M., Khan, A.A., Mehmood, R., Algarni, A., Albeshri, A., Katib, I.: A framework for faster porting of scientific applications between heterogeneous clouds. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, pp. 27–43. Springer, Cham (2018)
10. Alotaibi, S., Mehmood, R.: Big data enabled healthcare supply chain management: opportunities and challenges. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 207–215. Springer, Cham (2018)
11. Alamoudi, E., Mehmood, R., Albeshri, A., Gojobori, T.: DNA profiling methods and tools: a review. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 216–231. Springer, Cham (2018)
12. Al Shehri, W., Mehmood, R., Alayyaf, H.: A Smart pain management system using big data computing. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for

Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 232–246. Springer, Cham (2018)
13. Khanum, A., Alvi, A., Mehmood, R.: Towards a semantically enriched computational intelligence (SECI) framework for smart farming. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 247–257. Springer, Cham (2018)
14. Aqib, M., Mehmood, R., Albeshri, A., Alzahrani, A.: Disaster management in smart cities by forecasting traffic plan using deep learning and GPUs. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 139–154. Springer, Cham (2018)
15. Alam, F., Mehmood, R., Katib, I.: D2TFRS: an object recognition method for autonomous vehicles based on RGB and spatial values of pixels. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 155–168. Springer, Cham (2018)
16. Muhammed, T., Mehmood, R., Albeshri, A.: Enabling reliable and resilient IoT based smart city applications. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 169–184. Springer, Cham (2018)
17. Al-Dhubhani, R., Mehmood, R., Katib, I., Algarni, A.: Location privacy in smart cities era. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 123–138. Springer, Cham (2018)
18. Alomari, E., Mehmood, R.: Analysis of tweets in Arabic language for detection of road traffic conditions. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 98–110. Springer, Cham (2018)
19. Arfat, Y., Aqib, M., Mehmood, R., Albeshri, A., Katib, I., Albogami, N., Alzahrani, A.: Enabling smarter societies through mobile big data fogs and clouds. Procedia Comput. Sci. **109**, 1128–1133 (2017)
20. Schlingensiepen, J., Nemtanu, F., Mehmood, R., McCluskey, L.: Autonomic transport management systems—enabler for smart cities, personalized medicine, participation and industry grid/industry 4.0. In: Intelligent Transportation Systems—Problems and Perspectives, pp. 3–35. Springer, Cham (2016)
21. Alyahya, H., Mehmood, R., Katib, I.: Parallel sparse matrix vector multiplication on intel MIC: performance analysis. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 306–322. Springer, Cham (2018)
22. Arfat, Y., Mehmood, R., Albeshri, A.: Parallel shortest path graph computations of united states road network data on apache spark. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 323–336. Springer, Cham (2018)
23. Kruse, C.S., Goswamy, R., Raval, Y., Marawi, S.: Challenges and opportunities of big data in health care: a systematic review. JMIR Med. Inf. **4**, e38 (2016)
24. Sivarajah, U., Kamal, M.M., Irani, Z., Weerakkody, V.: Critical analysis of big data challenges and analytical methods. J. Bus. Res. **70**, 263–286 (2017)

25. Chauhan, S., Agarwal, N., Kar, A.K.: Addressing big data challenges in smart cities: a systematic literature review. Info. **18**, 73–90 (2016)
26. Chen, M., Mao, S., Liu, Y.: Big data: a survey. Mob. Netw. Appl. **19**, 171–209 (2014)
27. Padhy, R.P.: Big data processing with Hadoop-MapReduce in cloud systems. IJ-CLOSER Int. J. Cloud Comput. Serv. Sci. **2**, 233–245 (2012)
28. Singh, K., Kaur, R.: Hadoop: addressing challenges of big data. In: 2014 IEEE International Advance Computing Conference (IACC), pp. 686–689. IEEE (2014)
29. Xu, Z., Shi, Y.: Exploring big data analysis: fundamental scientific problems. Ann. Data Sci. **2**(4), 363–372 (2015)
30. Hashem, I.A.T., Yaqoob, I., Badrul Anuar, N., Mokhtar, S., Gani, A., Ullah Khan, S.: The rise of "Big Data" on cloud computing: review and open research issues. Inf. Syst. **47**, 98–115 (2014)
31. Radha, K., Rao, B.T.: Slot utilization and performance improvement in hadoop cluster. Presented at the (2016)
32. Guo, Z., Fox, G., Zhou, M.: Investigation of data locality in MapReduce. In: IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), pp. 419–426. IEEE (2012)
33. Eltabakh, M.Y., Tian, Y., Özcan, F., Gemulla, R., Krettek, A., McPherson, J.: CoHadoop: flexible data placement and its exploitation in Hadoop. Proc. VLDB Endow. **4**, 575–585 (2011)
34. Wang, L., Tao, J., Ranjan, R., Marten, H., Streit, A., Chen, J., Chen, D.: G-Hadoop: MapReduce across distributed data centers for data-intensive computing. Futur. Gener. Comput. Syst. **29**, 739–750 (2013)
35. Hsu, C.-H., Slagter, K.D., Chung, Y.-C.: Locality and loading aware virtual machine mapping techniques for optimizing communications in MapReduce applications. Futur. Gener. Comput. Syst. **53**, 43–54 (2015)
36. Yu, X., Hong, B.: Grouping blocks for MapReduce co-locality. In: 2015 IEEE International Parallel and Distributed Processing Symposium, pp. 271–280. IEEE (2015)
37. Lin, Z., Cai, M., Huang, Z., Lai, Y.: SALA: a skew-avoiding and locality-aware algorithm for MapReduce-Based Join. 1, 311–323 (2014)
38. Rhine, R., Bhuvan, N.T.: Locality Aware MapReduce, pp. 221–228. Springer, Cham (2016)
39. Chen, T.Y., Wei, H.W., Wei, M.F., Chen, Y.J., Hsu, T.S., Shih, W.K.: LaSA: a locality-aware scheduling algorithm for Hadoop-MapReduce resource assignment. Proc. 2013 Int. Conf. Collab. Technol. Syst. CTS 2013, pp. 342–346 (2013)
40. Tan, J., Meng, S., Meng, X., Zhang, L.: Improving reducetask data locality for sequential MapReduce jobs. In: 2013 Proceedings IEEE INFOCOM, pp. 1627–1635. IEEE (2013)
41. Ibrahim, S., Jin, H., Lu, L., Wu, S., He, B., Qi, L.: LEEN: locality/fairness-aware key partitioning for MapReduce in the Cloud. In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science, pp. 17–24. IEEE (2010)
42. Panchputre, K., Chaudhary, P., Garg, R.: Locality-aware load balancer for HBase, pp. 1–8
43. Wang, K., Zhou, X., Li, T., Zhao, D., Lang, M., Raicu, I.: Optimizing load balancing and data-locality with data-aware scheduling. In: Proceedings—2014 IEEE International Conference on Big Data, IEEE Big Data 2014, pp. 119–128 (2015)
44. Park, J., Lee, D., Kim, B., Huh, J., Maeng, S.: Locality-aware dynamic VM reconfiguration on MapReduce clouds. In: Proceedings of the 21st International Symposium on High-Performance Parallel and Distributed Computing—HPDC'12, pp. 27. ACM Press, New York (2012)
45. Zhang, X., Feng, Y., Feng, S., Fan, J., Ming, Z.: An effective data locality aware task scheduling method for MapReduce framework in heterogeneous environments. In: 2011 International Conference on Cloud and Service Computing, pp. 235–242. IEEE (2011)
46. Fan, X., Ma, X., Liu, J., Li, D.: Dependency-aware data locality for MapReduce. In: IEEE International Conference on Cloud Computing CLOUD, pp. 408–415 (2014)
47. Khan, M., Liu, Y., Li, M.: Data locality in Hadoop cluster systems. In: 2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2014), pp. 720–724 (2014)

48. Chen, Y., Liu, Z., Wang, T., Wang, L.: Load balancing in MapReduce based on data locality. Presented at the (2014)
49. Kc, K., Freeh, V.W.: Dynamically controlling node-level parallelism in Hadoop. 2015 IEEE 8th Int. Conf. Cloud Comput., pp. 309–316 (2015)
50. Palit, I., Reddy, C.K.: Scalable and parallel boosting with mapReduce. IEEE Trans. Knowl. Data Eng. **24**, 1904–1916 (2012)
51. Perkins, L.S., Andrews, P., Panda, D., Morton, D., Bonica, R., Werstiuk, N., Kreiser, R.: A survey of load balancing techniques for data intensive computing. In: 2009 International Symposium on Collaborative Technologies and Systems (CTS 2009), vol. 41, p. c1 (2011)
52. Ajitha, A., Ramesh, D.: Improved task graph-based parallel data processing for dynamic resource allocation in cloud. Procedia Eng. **38**, 2172–2178 (2012)
53. Nishanth, S., Radhikaa, B., Ragavendar, T.J., Babu, C., Prabavathy, B.: CoHadoop ++ : a load balanced data colocation in radoop distributed file system. In: Proceedings of 2013 5th International Conference on Advanced Computing, pp. 100–105 (2013)
54. Xu, Y., Qu, W., Li, Z., Liu, Z., Ji, C., Li, Y., Li, H.: Balancing reducer workload for skewed data using sampling. Comput. Electr. Eng. **40**, 675–687 (2014)
55. Chen, Q., Yao, J., Xiao, Z.: LIBRA: Lightweight Data Skew Mitigation in MapReduce. IEEE Trans. Parallel Distrib. Syst. **9219**, 1–1 (2014)
56. Zhou, H., Wen, Q.: Load balancing solution based on AHP for Hadoop. In: 2014 IEEE Workshop on Electronics, Computer and Applications pp. 633–636 (2014)
57. Gao, Z., Liu, D., Yang, Y., Zheng, J., Hao, Y.: A load balance algorithm based on nodes performance in Hadoop cluster. In: APNOMS 2014—16th Asia-Pacific Network Operations and Management Symposium, pp. 1–4 (2014)
58. Fadika, Z., Dede, E., Hartog, J., Govindaraju, M.: MARLA: MapReduce for heterogeneous clusters. In: Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012), pp. 49–56. 2012.eneous clusters (2012)
59. Wang, Y., Croft, W.L.: Smart shuffling in MapReduce: a solution to Balance Network Traffic and Workloads (2015)
60. Myung, J., Shim, J., Yeon, J., Lee, S.: Handling data skew in join algorithms using MapReduce. Expert Syst. Appl. **51**, 286–299 (2016)
61. Xie, J.X.J., Yin, S.Y.S., Ruan, X.R.X., Ding, Z.D.Z., Tian, Y.T.Y., Majors, J., Manzanares, A., Qin, X.Q.X.: Improving MapReduce performance through data placement in heterogeneous Hadoop clusters. In: 2010 IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum (IPDPSW), vol. 9, pp. 29–42 (2010)
62. Arasanal, R.M., Rumani, D.U.: Improving MapReduce performance through complexity and performance based data placement in heterogeneous hadoop clusters. In: Presented at the (2013)
63. Lee, C.W., Hsieh, K.Y., Hsieh, S.Y., Hsiao, H.C.: A dynamic data placement strategy for Hadoop in heterogeneous environments. Big Data Res. **1**, 14–22 (2014)
64. Sujitha, S., Jaganathan, S.: Aggrandizing Hadoop in terms of node heterogeneity & data locality. In: 2013 IEEE International Conference on Smart Structures and Systems, ICSSS 2013, 145–151 (2013)
65. Ubarhande, V., Popescu, A.-M., Gonzalez-Velez, H.: Novel data-distribution technique for Hadoop in heterogeneous cloud environments. In: 2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems, pp. 217–224. IEEE (2015)
66. Huang, X., Zhang, L., Li, R., Wan, L., Li, K.: Novel heuristic speculative execution strategies in heterogeneous distributed environments. Comput. Electr. Eng. **50**, 166–179 (2015)
67. Prasad, M.S.G., Nagesh, H.R., Prabhu, S.: Performance analysis of schedulers to handle multi jobs in Hadoop cluster. Int. J. Mod. Educ. Comput. Sci. **7**, 51–56 (2015)
68. Sethi, K.K., Ramesh, D.: Delay scheduling with reduced workload on JobTracker in Hadoop. Presented at the (2016)
69. Zaharia, M., Borthakur, D., Sarma, J. S., Elmeleegy, K., Shenker, S., Stoica, I.: Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In: Proceedings of the 5th European conference on Computer systems—EuroSys '10, 2010, p. 265.

70. Sun, M., Zhuang, H., Li, C., Lu, K., Zhou, X.: Scheduling algorithm based on prefetching in MapReduce clusters. Appl. Soft Comput. **38**, 1–10 (2015)
71. Gu, R., Yang, X., Yan, J., Sun, Y., Wang, B., Yuan, C., Huang, Y.: SHadoop: improving MapReduce performance by optimizing job execution mechanism in Hadoop clusters. J. Parallel Distrib. Comput. **74**, 2166–2179 (2014)
72. Yang, Y., Xu, J., Wang, F., Ma, Z., Wang, J., Li, L.: A MapReduce task scheduling algorithm for deadline-constraint in homogeneous environment. In: 2014 Second International Conference on Advanced Cloud and Big Data, pp. 208–212. IEEE (2014)
73. Sadasivam, G.S., Selvaraj, D.: A novel parallel hybrid PSO-GA using MapReduce to schedule jobs in Hadoop data grids. In: Proceedings—2010 Second World Congress Nature and Biologically Inspired Computing NaBIC 2010, pp. 377–382 (2010)
74. Li, L., Tang, Z., Li, R., Yang, L.: New improvement of the Hadoop relevant data locality scheduling algorithm based on LATE. In: Procedings of 2011 International Conference on Mechatron Science, Electric Engineering and Computer, MEC 2011, pp. 1419–1422 (2011)
75. Zaharia, M., Chowdhury, M., Das, T., Dave, A.: Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, NSDI'12, pp. 2–2 (2012)
76. Engle, C., Lupher, A., Xin, R., Zaharia, M., Franklin, M.J., Shenker, S., Stoica, I.: Shark: fast data analysis using coarse-grained distributed memory. In: Proceedings of the SIGMOD—International Conference on Management of Data, pp. 689–692 (2012)
77. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. https://dl.acm.org/citation.cfm?id=1863103.1863113 (2010)
78. Dokeroglu, T., Ozal, S., Bayir, M.A., Cinar, M.S., Cosar, A.: Improving the performance of Hadoop Hive by sharing scan and computation tasks. J. Cloud Comput. **3**, 12 (2014)
79. He, Y., Lee, R., Huai, Y., Shao, Z., Jain, N., Zhang, X., Xu, Z.: RCFile: A fast and space-efficient data placement structure in MapReduce-based warehouse systems. In: Proceedings of the International Conference on Data Engineering, pp. 1199–1208 (2011)
80. Thusoo, A., et al.: Hive—a petabyte scale data warehouse using Hadoop. In: Proceedings of the ICDE, pp. 996–1005 (2010)
81. Dokeroglu, T., Cınar, M.S., Yazıcı, A., Sert, S.A., Cosar, A.: Improving Hadoop hive query response times through efficient virtual resource allocation. Flex. Query Ans. Syst. **5822**, 88–98 (2009)
82. Xin, R.S., Rosen, J., Zaharia, M., Franklin, M.J., Shenker, S., Stoica, I.: Shark:SQL and rich analytics at scale. In: Proceedings of the 2013 International Conference on Management of data, SIGMOD'13, pp. 13–24 (2013)
83. Wang, G., Chan, C.-Y.: Multi-query optimization in MapReduce framework. In: Proceedings of VLDB Endowment, pp. 145–156 (2013)
84. Bissiriou, C.A.A., Chaoui, H.: Big data analysis and query optimization improve HadoopDB performance. In: Proceedings of the 10th International Conference on Semantic Systems, SEM'14, pp. 1–4 (2014)
85. Silva, Y.N., Reed, J.M.: Exploiting MapReduce-based similarity joins. In: Proceedings of the 2012 International Conference on Management Data—SIGMOD'12, vol. 693 (2012)
86. Suciu, D.: Distributed query evaluation on semistructured data. ACM Trans. Database Syst. **27**, 1–62 (2002)
87. Ding, D., Dong, F., Luo, J.: Multi-Q: multiple queries optimization based on MapReduce in cloud. In: 2014 Second International Conference on Advanced Cloud and Big Data, pp. 100–107 (2014)
88. Aly, A.M., Elmeleegy, H., Qi, Y., Aref, W.: Kangaroo. In Proceedings of the Ninth ACM International Conference on Web Search Data Mining—WSDM'16, pp. 397–406 (2016)
89. Zou, H., Yu, Y., Tang, W., Chen, H.W.M.: FlexAnalytics: A flexible data analytics framework for big data applications with I/O performance improvement. Big Data Res. **1**, 4–13 (2014)
90. Li, H., Ghodsi, A., Zaharia, M., Baldeschwieler, E., Shenker, S., Stoica, I.: Tachyon: memory throughput I/O for cluster computing frameworks. Memory. **18**, 1 (2013)

91. Yu, W., Member, S., Wang, Y., Que, X., Xu, C.: Virtual shuffling for efficient data movement in MapReduce. IEEE Trans. Comput. **64**, 556–568 (2015)
92. Yin, J., Wang, J.: Optimize parallel data access in big data processing. In: 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 721–724 (2015)
93. Wang, J., Xiao, Q., Yin, J., Shang, P.: DRAW: a new Data-gRouping-AWare data placement scheme for data intensive applications with interest locality. IEEE Trans. Magn. **49**, 2514–2520 (2013)
94. Xue, R., Gao, S., Ao, L., Guan, Z.: BOLAS: bipartite-graph oriented locality-aware scheduling for MapReduce tasks. In: 2015 14th International Symposium on Parallel and Distributed Computing, pp. 37–45. IEEE (2015)
95. Satapathy, S.C., Mandal, J.K., Udgata, S.K., Bhateja, V.: Information systems design and intelligent applications, vol. 434. Springer, New Delhi (2016)
96. Tung, L.-D., Nguyen-Van, Q., Hu, Z.: Efficient query evaluation on distributed graphs with hadoop environment. In: ACM International Conference Proceedings Series, pp. 311–319 (2013)

Chapter 21
Software Quality in the Era of Big Data, IoT and Smart Cities

Fatmah Yousef Assiri and Rashid Mehmood

21.1 Introduction

Software quality is the degree to which the software conforms to its requirements. General software quality attributes include testability, maintainability, efficiency, and reliability. One important aspect of software quality is software correctness, which concerns how well the program provides the required functionalities, as defined by its specifications, and can be achieved through software testing and debugging. Software testing is a dynamic process that executes the software under study using a set of test inputs to ensure its outputs meet the users' expectations. If the software behavior fails to perform as expected, software debugging is performed, which involves checking the code to determine the cause of failures and fixing them.

Software testing and debugging are time-consuming. Studies show that software debugging and testing form between 50 and 70% of the total development cycle [41]. Software testing involves comparing a set of test inputs and expected results to the actual software outputs. If the software outputs fail to match the expected ones, a fault is detected and the software must be checked for errors. Code is debugged to locate faults and fix them. As requirements change, the software is tested again to ensure that it continues to return the expected behavior, and additional tests are written to test any new requirements; however, writing new tests is not a trivial process.

F. Y. Assiri (✉)
College of Computer Science and Engineering, University of Jeddah, Jeddah, Saudi Arabia
e-mail: fyassiri@uj.edu.sa

R. Mehmood
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: RMehmood@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_21

The complexity of software is on the rise with the developments of *smart cities*. Smart cities are driven by, or involve, integration of multiple city systems, such as transport and healthcare, with the aim to provide its citizens a high quality of life [76], see, e.g., [72] for motivations of smart cities and societies. Integrating multiple complex systems causes an increase in the complexity of the underlying software interactions and leads to a higher software complexity. This in turn makes the software quality a bigger challenge.

Relatedly, big data and Internet of Things (IoT) are driving radical changes in smart cities designs, and hence, the software systems landscape. *Big data* "refers to the emerging technologies that are designed to extract value from data having four Vs characteristics; volume, variety, velocity and veracity [71]." The Internet of Things (IoT) becomes one of the key technological developments of our times that we are able to realize its full potential; it is expected to be a major producer of big data [5]. IoT is defined as "a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies [81]."

Together, big data, IoT, smart cities, and other emerging complex applications have exacerbated the challenges of maintaining software quality. The big data produced by IoT and other sources is used in designing or operating various software machines and systems. Since the data is uncertain (i.e., the veracity characteristic), it could lead to inaccurate or faulty system behavior. For example, a computed tomography (CT) scan based on inaccurate machine behavior, or inaccurate data, may give a false positive result for cancer. A wearable device may analyze the data of a diabetic patient incorrectly, giving false negative results, leading to no insulin dose for a patient who actually needed a high dose of insulin. Automatic surgery machines, autonomous vehicles, and spaceships all are examples of critical software with high software and data quality requirements. Moreover, data is being used by organizations to develop strategies, policies, and operations; inaccurate data could lead to disastrous outcomes for these organizations and even for the whole national or global economy.

The aim of this paper is to review the technologies related to software quality in the era of big data, IoT, and smart cities. We elaborate on software quality processes, software testing and debugging. Model checking is discussed with some thoughts on the role it could play in the big data era and the benefits it could gain from big data. The role of big data in software quality is explored. Conclusion is drawn to suggest future directions.

The remainder of the paper is structured as follows. Section 21.2 discusses software quality, software testing and debugging. Section 21.3 discusses model checking. Section 21.4 introduces big data and reviews some related work. Section 21.5 presents a review of the work that applies data mining techniques to utilize available data to improve software quality. Section 21.6 concludes the paper.

21.2 Software Quality

Software quality is the degree to which the software conforms to a set of requirements that meet the design specification and the users' expectations. Quality can be viewed and evaluated from the aspects of function, structure, and process [26]. Functional quality concerns the conformance of the tasks to the users' required functionalities, with few defects as possible. Structural quality relates to the quality of the written code and can be measured by code maintainability, testability, and understandability. Process quality relates to the development process such as meeting the delivery deadlines and budgets. These three aspects of software quality interleave and thus affect each other.

Software testing and debugging are among the main activities in the development cycle that guarantee the quality of the developed software. Software testing is a validation process that is conducted to ensure that the software meets its specifications, and software debugging is the process of analyzing the code to locate errors that caused the software to fail and correcting them [41]. In Sects. 21.2.1 and 21.2.2, we explain the work that has been done in both areas.

21.2.1 Software Testing

Testing, which is among the main steps in the software development life cycle to ensure software quality, involves executing a set of input values and checking their outputs to validate that the software meets its requirements and intended usage[10]. Testing is a dynamic process performed by observing the software execution. If the resulting output differs from the expected results, a fault is detected. The process of finding these faults and correcting them is called *debugging*.

Testing can be done at different levels depending on the phase that has been performed. *Unit testing* evaluates the software at the implementation phase and tests each unit separately. Units can be an individual element of the software such as a method or a class. *System and integration testing* are performed when the system is complete. System testing verifies that the whole system meets the design specifications, and integration testing checks that the subsystems (group of units) integrate correctly.

Software testing is divided into *black-box* and *white-box* testing. Black-box testing examines the application functionalities without looking to internal structures. Black-box testing creates tests from the software requirements and specifications; one form of applying it is through the *equivalence class partitioning* in which the program behaves the same for each set of input values; each set is called a *class*. For example, the program should retain the same output values for all positive number, thus the set of positive number is considered a class, and the program should be tested with exactly one value of each class.

White-box testing (also known as structural testing) is a method of testing software functionalities (internal structure), and it can be applied through unit and system testing. Tests performed by the software development team are called *alpha testing*, and those performed by the customer are called *beta testing*. Beta testing is also a form of black-box testing [79].

Tests consist of a set of *test cases*. Each test case consists of input values and a *test oracle*, which compares the expected output with the actual output to determine whether a program has failed or not [20]. To overcome the problem of having no oracles or the time-consuming process of writing them [94], *metamorphic testing* was introduced [28, 97]. Metamorphic testing creates follow-up test cases from a set of initial test cases using metamorphic relations. For example, if the initial test evaluates the power function $f(x) = e^x$ and the value of x is (3), then e^2 is equal to value (let's assume its (8)). Metamorphic testing creates another test case which is the value of a is (-2), and the output is $(1/8)$. The metamorphic relation (MR) is used to check the outputs of the two tests. In this case, MR is that output of first test case (8) + the output of the second test case (1/8) is equal to (1). If MR does not satisfy, a failure is detected.

Mutation testing is an alternative testing approach which was designed to assess the quality of the test cases [35, 46]. Mutation testing creates a copy of the original program, called a *mutant*, with a seeded fault. The faults are a simple syntax change injected to the code [61, 80]. Tests are executed and the fault is detected if the output of the mutant is different from the output of the original program. Mutation testing computes a *mutation adequacy score*, which represents the number of detected faults over the total number of seeded faults. A higher score indicates a higher quality of the test sets. *MuJava* tool was developed to perform automated mutation testing by generating mutants and computing the adequacy score for a set of JUnit tests [62].

Software testing is labor intensive; thus, to reduce the costs, many automation techniques were developed to automate the generation of test data and test oracles [22, 23, 36, 55, 74, 90].

21.2.2 Software Debugging

Software *debugging* is a diagnosis process for locating and fixing errors that cause software to fail. Fault localization (FL) techniques were introduced to locate statements in source code that are more likely to contain faults. FL computes a *suspiciousness score* for each statement, and the computed score indicates the probability that a statement contains a fault.

Spectrum-based FL (SBFL) [1, 4, 18, 29, 32, 49, 86], which is a common FL approach, is a dynamic process that counts the number of passed and failed tests executed for each statement and computes a suspiciousness score for each statement. Statements executed during a failed run are considered to be more likely to contain faults and are thus assigned a higher suspiciousness score than other statements.

Table 21.1 The dynamic behavior of the faulty program *gcd* when executed against tests in $T1$, . . ., $T5$. *Sus. Score* is the suspiciousness score computed using Tarantula

Stmt ID	Stmt	T_1	T_2	T_3	T_4	T_5	Sus. Score
	gcd (int a, int b) {						
1	if(a < 0) //fault					x	1.00
2	{ printf("%g \n", b);						0.00
3	return 0 ; }						0.00
4	while(b ! = 0)	x	x	x	x	x	0.50
5	if(a > b)	x	x		x	x	0.57
6	a = a − b ;		x		x		0.00
7	else	x	x		x	x	0.57
8	b = b − a ;	x	x		x	x	0.57
9	printf("%g \n", a) ;	x	x	x	x		0.00
10	return 0 ;	x	x	x	x		0.00
	}						

Many heuristics have been proposed to compute statement suspiciousness scores [1, 4, 48, 49, 77, 86].

To illustrate how FL techniques order statements based on the likelihood they contain faults, we used the C program shown in Table 21.1 that is adapted from [47]. The program computes the Euclid's greatest common divisor. This example used four passed tests: T1, T2, T3, and T4, and one failed test: T5. To compute the suspiciousness score, we applied the Tarantula heuristic (Eq. (21.1)). To reduce the time of performing this step, many tools have been developed to automate other parts of testing, such as the FL techniques [45, 47, 83].

$$susp\_Turantula(s) = \frac{\%FailedTests(s)}{\%PassedTests(s) + \%FailedTests(s)} \tag{21.1}$$

The debugging process also involves fixing located faults. Although this was traditionally a manual process, automated program repair (APR) techniques were developed to automate the process [52, 53, 59, 63, 78]. APR techniques take a faulty program and conduct a set of repair tests to produce a repaired program. Figure 21.1 describes the overall structure of the APR techniques. The APR technique applies an FL technique to create a list of potentially faulty statement (LPFS) that is ordered based on their likelihood of containing fault, creates a copy of the original program with one inserted change called a *variant*, and validates the created variant to check whether or not the fault is fixed.

To create the variants, a set of program modification operators (PMOs) are applied to change the code in the faulty statement generating the variant. PMOs are selected randomly or in order based on the applied search algorithm. Then, each variant is validated by executing it on a set of test cases, regression tests, or formal specifications. The variant is considered a *potential repair* or *potential repaired program* if it passes all the tests used in the process. The generated repair

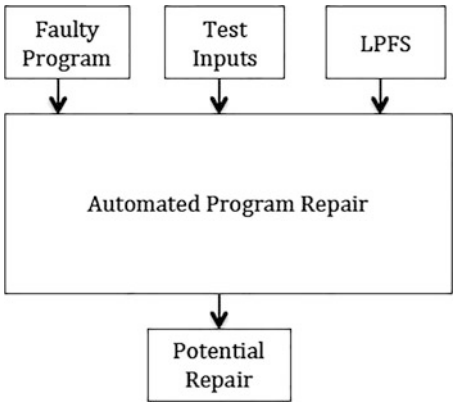


Fig. 21.1 Overall automated program repair (APR) technique adapted from [15]

is considered a potential repair, rather than a validated repair, because it is a repair with respect to the selected set of tests used in the process of fixing the faults. The repair is only considered a *valid repair* when it passes a set of tests (often regression tests) that were not included in the repair process.

Many researchers have contributed to improve the APR process and the quality of generate repairs. Debroy and Wong [33, 34] proposed using mutations through a brute-force search and an FL technique to automate fault fixing. Nguyen et al. [78] developed *SemFix*, which is a tool that locates faults using the Tarantula heuristic [49]. Then, symbolic execution and program synthesis were used to fix faults. Program syntheses are applied in a predefined order. Wei et al. [91] fix faults using Eiffel programs equipped with contracts, and Kim et al. [53] repaired faults by creating fix templates using 10 built-in patterns that were developed based on common patches written by humans. Weimer et al. [92] developed a weighting scheme to locate faults and applied an evolutionary algorithm to fix faults. APR techniques are also used to fix faults for executable software [25, 82]. Evolutionary computing and genetic programming have been adapted to repair faults in C software [38, 59, 92, 93], Java [12, 52], and Python [2], and to help satisfy non-functional requirements [13, 95].

The state-of-the-art APR technique is GenProg tool, which uses genetic programming to modify a program until it finds a variant that passes all the repair test [38, 59, 92, 93]. GenProg was used to successfully fix the Microsoft Zune bug date error, which froze Microsoft devices in 2008 due to an infinite loop that occurred on the last day of a leap year [75]. However, repairs generated using GenProg were hard to read and it only performed potential repairs since they failed when they were executed on a set of regression tests. Assiri and Bieman [15–17] proposed using first-order mutations with a stochastic search algorithm to generate repairs that are similar to efficient ones written by humans.

Even though debugging activities (locating and fixing faults) have been automated to reduce debugging costs, there are many new challenges particularly with big data because it runs largely on parallel cloud computing platforms, making

it error prone and inefficient. Researchers have developed debugging tools to overcome these problems.

BigDebug is an interactive debugging tool that allows developers to set breakpoints to inspect program states during program execution [40]. BigDebug also provides guarded watchpoints, which return a set of records that satisfy a given condition. BigDebug, which provides backward and forward tracking and allows developers to fix faults and resume execution, improves the performance, avoids having to start the execution from the beginning, and reduces the locations should be checked for failures.

Considerable research has developed debugging tools for distributed systems. However, these typically depend on the use of a single frontend that controls many backend debuggers, which slows the process when used for large-scale distributed systems. Mehmood et al. [70] improved the structure of debuggers to scale them to large systems. The proposed debugging tool follows a hierarchical approach by using intermediate backend servers for a limited number of processes (Fig. 21.2), which evaluate assertions on the connected processes and report violations. This method improves the FL and system overall traffic, making it a suitable approach for large-scale distributed systems.

An alternative method for debugging a distributed system is to perform the debugging at higher-abstraction level than the unit level [21]. When performed at the system level, system behavior is translated into a set of events that are filtered to remove all events that are not of interest to the user. Event sequences are then clustered to create one single event that is used to identify the cause of failures in complex distributed systems. Event definition language (EDL) is used to define a set of events based on a combination of previously determined events. Events are compiled and interpreted to determine the cause of the failures.

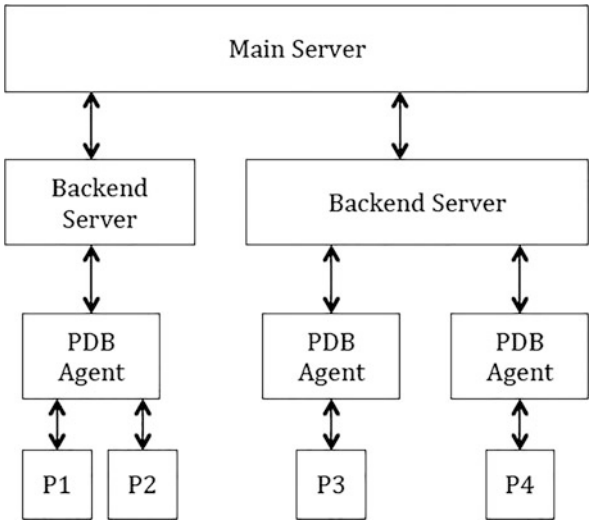


Fig. 21.2 PDB architecture adapted from [70]

Debugging tools rely on setting breakpoints or sets of slices to check the software's behavior. Thus, if the specified locations of the variables do not contain the cause of the errors, the tools will be unable to identify the faulty code. Andrew and Myers developed the *Whyline* tool [54], an interactive debugging tool that allows developers to ask questions for a given output. Whyline records execution traces for each event and each execution trace has a specific trace file. Then, an output history is created for all stored events. When a class is loaded, Whyline runs an algorithm that depends on data dependencies to identify all variables and fields affected by the output. After identifying the codes responsible for the specified output, the tool generates questions using static and dynamic methods. Two questions are asked: *why did* and *why did not*. The first question is answered using the dynamic slicing technique and the latter is answered by investigating each instruction individually. The evaluation study found that using Whyline improved the debugging time for novice programmers, but it suffers from performance issues.

21.3 Model Checking

Model checking is a verification method that is performed to ensure program correctness by investigating all possible software internal states. Model checking requires a complete and clear set of properties that describes what the system should and should not do. The software states are checked against the specified properties. If a violation is found, counterexamples to the execution paths that caused the violation are generated. Model checking has been used to debug many systems such as airline reservation and e-commerce systems [19].

Model checking has also been used to automate software testing (see Callahan et al. [24]). White-box testing, which concerns the software's internal representation through the investigation of execution traces for intermediate values, detects errors if an inconsistency exists between the actual and expected values. Specification-based testing, which uses model checking techniques, was proposed to validate and generate tests during the software evolutionary process. In this method, a computation tree comprising all possible execution paths is generated and searched to ensure that all paths follow the specified constraints.

Even though the work by Callahan et al. [24] used a model checker to generate test cases automatically, Amman et al. [9, 11] proposed using a model checker to generate mutation-adequate test cases by adapting mutation testing. Model checking is used widely to write and validate specifications. The proposed combination of model checking and mutation testing addresses the limitation of automatic test generation and mutation testing at the system level. System specifications are converted into a format used by the model checker using a modeling tool. Then, the generated specifications are mutated and used by the model checker to create counterexamples, which are used to automatically generate test cases. Tests are executed and the results and coverage are reported.

For test generation, the SPIN model checker [44] is used to identify execution trace paths for a specified property. Paths are validated and divided into partitions based on a defined set of requirements; each partition, which is called a *coverage property*, consists of a set of execution paths. Test templates, comprising actual test sequences, are generated using SPIN and are used to create invalid coverage properties to force the program to fail.

Formal methods, such as software cost reduction (SCR), have been used to improve software quality. SCR reduces the development cost since it helps to detect violations at an early stage in the software life cycle before the implementation [39]. SCR uses requirements to generate test sequences that consist of a set of input values and a set of output values for each input. The input values are validated by checking the set of constraints that are specified through the requirement specifications. Then, the test sequences are divided into equivalent partitions and test inputs are generated for all partitions.

Model checking relies on building models of the actual systems and then verifying the models, and therefore, big data technologies can be used to automate the process of model building. Big data technologies could also improve the quality of models that are built before being model checked. Alternatively, model checking can be applied to address the veracity challenges of big data.

While model checking has been very successful in verifying real-life systems, its biggest hurdle is the state-space explosion problem. Researchers have developed various techniques to address this challenge. These include, among others, the use of high performance computing techniques, see, e.g., [66, 67, 69].

21.4 Big Data

Big data is a relatively new research area that has been utilized in many fields such as online retail stores, decision-making, and scientific research [27]. Big data is defined variously in the literature: some researchers define it using the 3Vs: volume, velocity, and variety [56]. Volume relates to the size of the data, velocity is the speed of the data stream, and variety refers to the data types. Other researchers define big data using 4Vs, with the forth V referring to value, variability, or virtual [98]. Fen and Befit defined big data as the 3Vs plus two more: variability (data interpretation) and value (making decisions) [37]. We consider the definition where volume, variety, velocity, and veracity are used as the 4Vs of big data [71], and consider veracity, as many have noted, to be the biggest challenge of big data.

Big data applications can be used in business, technology, health, and smart cities. Big data can be used to improve quality of life. Data have been used in online retail stores, such as Amazon, to identify user preferences. Algorithms collect information about the users' preferences based on their actions [65]. In addition, the amount of healthcare data is increasing and is expected to reach a zettabyte in the near future in the USA [85]. Using this medical data will benefit individuals' health by enabling doctors to detect diseases at the early stages and determine treatments,

recovery options, and risks. For additional works on big data in context of smart cities, see [6, 7, 14, 68, 73, 88].

21.5 Big Data and Software Quality

Data can be used as a validity tool to ensure software correctness, build recommender systems, and predict future actions. Big data has been utilized in many sectors such as healthcare, banking, and transportation. Data are processed using data mining techniques to determine trends and to help in decision-making. Software quality can be related to big data in at least two ways. Firstly, big data can help develop better software quality techniques. Secondly, software quality techniques are needed to improve the quality of big data software and possibly deal with the big data veracity challenge.

With respect to software quality, existing work has applied data mining techniques to analyze data repositories, fix faults, determine trends, and automate test generation.

21.5.1 Mining Big Data

Data mining is performed to analyze large amounts of data to understand trends in the data and support decision-making [42]. Software intelligence (SI) is a new field of mining software data to help practitioners in daily decision-making processes, such as when to release the system, what part of the system to test, and/or what part to change [43].

Mining software repositories is a research direction that analyzes data repositories to obtain useful information about systems and projects. The types of repositories include historical repositories that show project progress; run-time repositories, which show system usage on deployment sites; and code repositories, which contain the code for software versions. Linking code repositories and bug repositories can provide a method for warning practitioners about bugs and risky codes.

Lin and Ryaboy analyzed Twitter data using data mining tools; however, due to the limitations of existing tools, the analysis was not a straightforward process [60]. In [89], the researchers mined heterogeneous information using the semantics of node types and the links between them in the networks. The researchers in [51] studied the potential of mining big graphs and found the PEGASUS tool to be a promising approach since it finds anomalous in the large Twitter connected graphs. Last, the authors in [8] focused on mining a large stream of Netflix Prize data to personalize recommendations. To improve the probabilities of customers selections, a lot of factors and more data need to be considered.

The authors in [50] used mining bug reports to develop the *BugMiner* tool, which uses the support vector machines (SVM) machine learning technique to perform a completion check and a redundancy check on new reports and estimate bug report trends (e.g., incident rate over time) of bug report databases using natural language processing. SVM used the historic reports to train the model to fill any missing fields. For any given report, the tool checks if it already exists by applying similarity ranking using cosine similarity, and Weibull distribution uses historic data to estimate the number of bug reports received during a specified period (weeks or months) after the start of the project. The experimental results showed that BugMiner was effective in terms of bug reports completion, redundancy, and finding trends. The authors suggest combining the tool with other bug tracking tools to create advanced intelligent software.

Mining software is also used to develop a repair model in the area of APR [64]. In their paper, the authors mine software repositories by investigating developers comments to generate repair actions that can be used later to fix faults. Repair actions can be in the form of adding a method call or changing the condition of *if* statements. Repair actions are then assigned different probabilities that are also learned from the repositories. To collect fixes from repositories, the authors used data set of 14 repositories and checked the differences between transitions at the abstract syntax tree (AST) level. A difference algorithm was used to produce the set of changes between each pair of Java files. The authors generated 41 change types and 137 possible change type entity types. The empirical study found that 28% of the changes were statement insertions, 23% were statement deletions, and 23% were statement updates. However, the change type *statement insert* was composed of many entity types, e.g., insert method invocation, *if* conditional, insert new variable. The results showed that the probability distribution of change type is project independent.

To repair faults, the authors of [64] created a repair model and used different approaches to compute the probabilities of each repair action. The repair shape, which is a set of all possible combinations of repair actions, was then created. The search space is a combination of fault space, repair shapes, and the concrete repair actions that create the shape.

In [96], the authors mined software repositories to study the co-evolution of the production code and test code. Repository histories and log messages were analyzed; however, the results found no matching between changes in the production code and the test. In other words, the test codes remained the same after changing the production code. The test coverage also dropped since no new test was created to guarantee the coverage of the new boundary values. Despite the notable finding, the study failed to specify which data mining techniques were used to check the repositories.

Data mining algorithms are used to automatically induce missing functional requirements from data executions [58]. This approach can help to recover missing and incomplete specifications, design regression tests, and evaluate the correctness of software. Creating up-to-date regression tests is difficult, especially with legacy systems. One way to create regression tests is to identify the input–output

relationships to write the requirements of the existing system. In [57], the authors proposed to identify the input–output relationships automatically using info-fuzzy networks (IFN), and they evaluated the effectiveness of IFN methodology on complex systems. The experimental results found that the data mining methods are effective for generating tests automatically without needing humans or complete sets of requirements since functional requirements are learned from data execution.

This study compares two approaches of automated construction of oracle: artificial neural networks (ANNs) and IFNs [3]. ANNs have been used to generate a minimal set of tests that are effective at revealing faults [57, 87]. To generate oracles automatically, the following three steps are performed: (1) the training phase, where the system is given positive oracles; (2) the evaluation phase, which accepts positive oracles and rejects negative ones; and (3) the decision phase in which the trained oracles identify correct test cases from unlabeled ones. The experimental results found that IFN would be more appropriate for testing applications that are at the early stages. However, ANNs appear to be better at identifying hard-to-detect faults.

Data mining techniques have been adapted to troubleshoot distributed systems [30]. The goal of this approach is to identify which resources properties would succeed or fail for specific jobs. To demonstrate this approach, the job and machine features for 1000 jobs were extracted, and the job status was described as either a success or failure. Then, two data mining techniques were applied to generate a prediction model: C4.5 decision tree [84] and RIPPER rule-based classification algorithm [31]. Even though both methods predicted that the same features would cause the failures, RIPPER was found to be a more robust and promising method. While other data mining techniques, such as the *lazy learning* technique, can be applied, they tend to require more information before drawing the model. Additional research is needed to examine more internal or external features.

21.6 Summary, Conclusions, and Future Work

Software quality is the degree to which the software conforms to its requirements. General software quality attributes include testability, maintainability, efficiency, and reliability. One important aspect of software quality is software correctness, which concerns how well the program provides the required functionalities, as defined by its specifications, and can be achieved through software testing and debugging. The complexity of software is on the rise with the developments of smart cities due to the complex nature of these applications and environments. Big data and Internet of Things (IoT) are driving radical changes in the software systems landscape. Together, big data, IoT, smart cities, and other emerging complex applications have exacerbated the challenges of maintaining software quality.

The big data produced by IoT and other sources is used in designing or operating various software machines and systems. Since the data is uncertain (i.e., the veracity characteristic), it could lead to inaccurate or faulty system behavior. In this paper, we reviewed the technologies related to software quality in the era of big data, IoT,

and smart cities. We elaborated on software quality processes, software testing and debugging. Model checking was discussed with some directions on the role it could play in the big data era and the benefits it could gain from big data. The role of big data in software quality was explored.

We discussed that software quality can be related to big data in at least two ways. Firstly, big data can help develop better software quality techniques. Secondly, software quality techniques are needed to improve the quality of big data software and possibly deal with the big data veracity challenge. We also highlighted that big data technologies can be used to automate the process of model building as part of the model checking process. Big data technologies could also improve the quality of models that are built before being model checked. Alternatively, model checking can be applied to address the veracity challenges of big data. As mentioned that the biggest hurdle of model checking is the state-space explosion problem that could be addressed using high performance computing techniques.

Our future work will focus on bringing together cutting-edge software quality and big data techniques to develop novel techniques for improving software and data quality of smart city systems.

References

1. Abreu, R., Zoeteweij, P., Van Gemund, A.J.: On the accuracy of spectrum-based fault localization. In: Testing: Academic and Industrial Conference Practice and Research Techniques-MUTATION, 2007. TAICPART-MUTATION 2007, pp. 89–98. IEEE, Piscataway (2007)
2. Ackling, T., Alexander, B., Grunert, I.: Evolving patches for software repair. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO '11, pp. 1427–1434. ACM, New York (2011)
3. Agarwal, D.: A comparative study of artificial neural networks and info fuzzy networks on their use in software testing. Master's Thesis, University of South Florida (2004)
4. Agrawal, H., Horgan, J.R., London, S., Wong, W.E.: Fault localization using execution slices and dataflow tests. In: Proceedings of the Sixth International Symposium on Software Reliability Engineering, pp. 143–151. IEEE, Piscataway (1995)
5. Alam, F., Mehmood, R., Katib, I., Albeshri, A.: Analysis of eight data mining algorithms for smarter internet of things (IOT). Procedia Comput. Sci. **98**, 437–442 (2016). https://doi.org/10.1016/j.procs.2016.09.068. http://www.sciencedirect.com/science/article/pii/S187705091632213X. The 7th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2016)/The 6th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2016)/Affiliated Workshops
6. Alomari, E., Mehmood, R.: Analysis of Tweets in Arabic Language for Detection of Road Traffic Conditions, pp. 98–110. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94180-6_12. http://link.springer.com/10.1007/978-3-319-94180-6_12
7. Alotaibi, S., Mehmood, R.: Big Data Enabled Healthcare Supply Chain Management: Opportunities and Challenges, pp. 207–215. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94180-6_21. http://link.springer.com/10.1007/978-3-319-94180-6_21
8. Amatriain, X.: Mining large streams of user data for personalized recommendations. ACM SIGKDD Explor. Newsl. **14**(2), 37–48 (2013)

9. Ammann, P.: System testing via mutation analysis of model checking specifications. ACM SIGSOFT Softw. Eng. Notes **25**(1), 33 (2000)
10. Ammann, P., Offutt, J.: Introduction to software testing, Cambridge University Press, Cambridge (2016)
11. Ammann, P.E., Black, P.E., Majurski, W.: Using model checking to generate tests from specifications. In: Proceedings of Second International Conference on Formal Engineering Methods, pp. 46–54. IEEE, Piscataway (1998)
12. Arcuri, A.: On the automation of fixing software bugs. In: Companion of the 30th International Conference on Software Engineering, ICSE Companion '08, pp. 1003–1006. ACM, New York (2008)
13. Arcuri, A., Yao, X.: A novel co-evolutionary approach to automatic software bug fixing. In: IEEE Congress on Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence), pp. 162–168. IEEE, Piscataway (2008)
14. Arfat, Y., Mehmood, R., Albeshri, A.: Parallel Shortest Path Graph Computations of United States Road Network Data on Apache Spark, pp. 323–336. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94180-6_30. http://link.springer.com/10.1007/978-3-319-94180-6_30
15. Assiri, F.Y., Bieman, J.M.: An assessment of the quality of automated program operator repair. In: Proceedings of the 2014 ICST Conference, ICST'14, IEEE, Piscataway (2014)
16. Assiri, F.Y., Bieman, J.M.: The impact of search algorithms in automated program repair. Submitted to the 2015 International Conference on Soft Computing and Software Engineering, (SeSe'15) (2015)
17. Assiri, F.Y., Bieman, J.M.: Fault localization for automated program repair: effectiveness, performance, repair correctness. Softw. Qual. J. **25**(1), 171–199 (2017)
18. Baah, G.K., Podgurski, A., Harrold, M.J.: The probabilistic program dependence graph and its application to fault diagnosis. IEEE Trans. Softw. Eng. **36**(4), 528–545 (2010)
19. Baier, C., Katoen, J.P.: Principles of model checking. MIT Press, Cambridge (2008)
20. Baresi, L., Young, M.: Test oracles. Tech. Rep., Technical Report CIS-TR-01-02, University of Oregon, Dept. of Computer and Information Science, Eugene, Oregon (2001)
21. Bates, P.C., Wileden, J.C.: High-level debugging of distributed systems: the behavioral abstraction approach. J. Syst. Softw. **3**(4), 255–264 (1983)
22. Boyapati, C., Khurshid, S., Marinov, D.: Korat: automated testing based on java predicates. In: ACM SIGSOFT Software Engineering Notes, vol. 27, pp. 123–133. ACM, New York (2002)
23. Burdonov, I., Kossatchev, A., Petrenko, A., Galter, D.: Kvest: automated generation of test suites from formal specifications. In: International Symposium on Formal Methods, pp. 608–621. Springer, Berlin (1999)
24. Callahan, J., Schneider, F., Easterbrook, S., et al.: Automated software testing using model-checking. In: Proceedings 1996 SPIN workshop, vol. 353 (1996)
25. Carzaniga, A., Gorla, A., Mattavelli, A., Perino, N., Pezze, M.: Automatic recovery from runtime failures. In: Proceedings of the 2013 International Conference on Software Engineering, pp. 782–791. IEEE, Piscataway (2013)
26. Chappell, D.: The three aspects of software quality: functional, structural, and process, White Paper. Chappell & Associates, San Francisco, CA. Available at www.davidchappell.com. Last accessed 30 May 2019
27. Chen, C.P., Zhang, C.Y.: Data-intensive applications, challenges, techniques and technologies: a survey on big data. Inf. Sci. **275**, 314–347 (2014)
28. Chen, T.Y., Cheung, S.C., Yiu, S.M.: Metamorphic testing: a new approach for generating next test cases. Tech. Rep., Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong (1998)
29. Chilimbi, T.M., Liblit, B., Mehra, K., Nori, A.V., Vaswani, K.: Holmes: effective statistical debugging via efficient path profiling. In: IEEE 31st International Conference on Software Engineering, 2009. ICSE 2009, pp. 34–44. IEEE, Piscataway (2009)

30. Cieslak, D.A., Thain, D., Chawla, N.V.: Short paper: troubleshooting distributed systems via data mining. In: 15th IEEE International Symposium on High Performance Distributed Computing, pp. 309–312. IEEE, Piscataway (2006)
31. Cohen, W.W.: Fast effective rule induction. In: Machine Learning Proceedings 1995, pp. 115–123. Elsevier, Amsterdam (1995)
32. Dallmeier, V., Lindig, C., Zeller, A.: Lightweight defect localization for Java. In: ECOOP 2005-Object-Oriented Programming, pp. 528–550. Springer, Berlin (2005)
33. Debroy, V., Wong, W.E.: Using mutation to automatically suggest fixes for faulty programs. In: Third International Conference on Software Testing, Verification and Validation (ICST), pp. 65–74. IEEE, Piscataway (2010)
34. Debroy, V., Wong, W.E.: Combining mutation and fault localization for automated program debugging. J. Syst. Softw. **90**, 45–60 (2014)
35. DeMillo, R.A., Lipton, R.J., Sayward, F.G.: Hints on test data selection: help for the practicing programmer. Computer **11**(4), 34–41 (1978)
36. Dick, J., Faivre, A.: Automating the generation and sequencing of test cases from model-based specifications. In: International Symposium of Formal Methods Europe, pp. 268–284. Springer, Berlin (1993)
37. Fan, W., Bifet, A.: Mining big data: current status, and forecast to the future. ACM SIGKDD Explor. Newsl. **14**(2), 1–5 (2013)
38. Forrest, S., Nguyen, T., Weimer, W., Le Goues, C.: A genetic programming approach to automated software repair. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, GECCO '09, pp. 947–954. ACM, New York (2009)
39. Gargantini, A., Heitmeyer, C.: Using model checking to generate tests from requirements specifications. In: ACM SIGSOFT Software Engineering Notes, vol. 24, pp. 146–162. Springer, Berlin (1999)
40. Gulzar, M.A., Interlandi, M., Yoo, S., Tetali, S.D., Condie, T., Millstein, T., Kim, M.: Bigdebug: debugging primitives for interactive big data processing in spark. In: Proceedings of the 38th International Conference on Software Engineering, pp. 784–795. ACM, New York (2016)
41. Hailpern, B., Santhanam, P.: Software debugging, testing, and verification. IBM Syst. J. **41**(1), 4–12 (2002)
42. Hand, D.J.: Principles of data mining. Drug Saf. **30**(7), 621–622 (2007)
43. Hassan, A.E., Xie, T.: Software intelligence: the future of mining software engineering data. In: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, pp. 161–166. ACM, New York (2010)
44. Holzmann, G.J.: Design and Verification of Computer Protocols, Prentice Hall, Upper Saddle River (1991)
45. Janssen, T., Abreu, R., van Gemund, A.J.: Zoltar: A toolset for automatic fault localization. In: Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering, pp. 662–664. IEEE Computer Society, Washington, D.C. (2009)
46. Jia, Y., Harman, M.: An analysis and survey of the development of mutation testing. IEEE Trans. Softw. Eng. **37**(5), 649–678 (2011)
47. Jones, J.A., Harrold, M.J.: Empirical evaluation of the Tarantula automatic fault-localization technique. In: Proceedings of the 20th IEEE/ACM international Conference on Automated Software Engineering, pp. 273–282. ACM, New York (2005)
48. Jones, J.A., Harrold, M.J., Stasko, J.T.: Visualization for fault localization. In: Proceedings of ICSE 2001 Workshop on Software Visualization, Toronto, Ontario, pp. 71–75. Citeseer (2001)
49. Jones, J.A., Harrold, M.J., Stasko, J.: Visualization of test information to assist fault localization. In: Proceedings of the 24th International Conference on Software Engineering, pp. 467–477. ACM, New York (2002)
50. Kaiser, L.W.B.X.G., Passonneau, R.: Bugminer: Software reliability analysis via data mining of bug reports. Delta **12**(10), 09–0500 (2011)
51. Kang, U., Faloutsos, C.: Big graph mining: algorithms and discoveries. ACM SIGKDD Explor. Newsl. **14**(2), 29–36 (2013)

52. Kern, C., Esparza, J.: Automatic error correction of Java programs. In: Proceedings of the 15th International Conference on Formal Methods for Industrial Critical Systems, FMICS'10, pp. 67–81. Springer, Berlin (2010)
53. Kim, D., Nam, J., Song, J., Kim, S.: Automatic patch generation learned from human-written patches. In: Proceedings of the 2013 International Conference on Software Engineering, pp. 802–811. IEEE, Piscataway (2013)
54. Ko, A.J., Myers, B.A.: Debugging reinvented: asking and answering why and why not questions about program behavior. In: Proceedings of the 30th International Conference on Software Engineering, pp. 301–310. ACM, New York (2008)
55. Lamancha, B.P., Polo, M., Caivano, D., Piattini, M., Visaggio, G.: Automated generation of test oracles using a model-driven approach. Inf. Softw. Technol. **55**(2), 301–319 (2013)
56. Laney, D.: 3d data management: controlling data volume, velocity and variety. META Group Res. Note **6**(70), 1 (2001)
57. Last, M., Kandel, A.: Automated test reduction using an info-fuzzy network. In: Software Engineering with Computational Intelligence, pp. 235–258. Springer, Boston (2003)
58. Last, M., Friedman, M., Kandel, A.: The data mining approach to automated software testing. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 388–396. ACM, New York (2003)
59. Le Goues, C., Nguyen, T., Forrest, S., Weimer, W.: GenProg: a generic method for automatic software repair. IEEE Trans. Softw. Eng. **38**(1), 54–72 (2012)
60. Lin, J., Ryaboy, D.: Scaling big data mining infrastructure: the twitter experience. ACM SIGKDD Explor. Newsl. **14**(2), 6–19 (2013)
61. Ma, Y.S., Kwon, Y.R., Offutt, J.: Inter-class mutation operators for java. In: Proceedings of 13th International Symposium on Software Reliability Engineering, 2002. ISSRE 2003, pp. 352–363. IEEE, Piscataway (2002)
62. Ma, Y.S., Offutt, J., Kwon, Y.R.: Mujava: a mutation system for Java. In: Proceedings of the 28th International Conference on Software Engineering, pp. 827–830. ACM, New York (2006)
63. Martinez, M., Monperrus, M.: Astor: evolutionary automatic software repair for Java. arXiv preprint arXiv:1410.6651 (2014)
64. Martinez, M., Monperrus, M.: Mining software repair models for reasoning on the search space of automated program fixing. Empir. Softw. Eng. **20**(1), 176–205 (2015)
65. McAfee, A., Brynjolfsson, E., Davenport, T.H., Patil, D., Barton, D.: Big data: the management revolution. Harv. Bus. Rev. **90**(10), 60–68 (2012)
66. Mehmood, R.: Disk-based techniques for efficient solution of large Markov chains. Ph.D. Thesis, School of Computer Science, University of Birmingham (2004)
67. Mehmood, R., Crowcroft, J.: Parallel iterative solution method for large sparse linear equation systems. Tech. Rep. UCAM-CL-TR-650, University of Cambridge, Computer Laboratory (2005). http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-650.pdf
68. Mehmood, R., Graham, G.: Big data logistics: a health-care transport capacity sharing model. Procedia Comput. Sci. **64**, 1107–1114 (2015). https://doi.org/10.1016/j.procs.2015.08.566. http://www.sciencedirect.com/science/article/pii/S1877050915027015. Conference on ENTERprise Information Systems/International Conference on Project MANagement/Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN / HCist 2015 October 7–9, 2015
69. Mehmood, R., Parker, D., Kwiatkowska, M.: An efficient BDD-based implementation of Gauss-Seidel for CTMC analysis. Tech Report, School of Computer Science, University of Birmingham (2003)
70. Mehmood, R., Crowcroft, J., Hand, S., Smith, S.: Grid-level computing needs pervasive debugging. In: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing, pp. 186–193. IEEE Computer Society, Washington, D.C. (2005)
71. Mehmood, R., Faisal, M.A., Altowaijri, S.: Future Networked Healthcare Systems: A Review and Case Study. In: Boucadair, M., Jacquenet, C. (eds.) Handbook of Research on Redesigning

the Future of Internet Architectures, pp. 531–558. IGI Global, Hershey (2015). https://doi.org/10.4018/978-1-4666-8371-6.ch022. http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-4666-8371-6.ch022
72. Mehmood, R., Alam, F., Albogami, N.N., Katib, I., Albeshri, A., Altowaijri, S.M.: Utilearn: A personalised ubiquitous teaching and learning system for smart societies. IEEE Access **5**, 2615–2635 (2017). https://doi.org/10.1109/ACCESS.2017.2668840
73. Mehmood, R., Meriton, R., Graham, G., Hennelly, P., Kumar, M.: Exploring the influence of big data on city transport operations: a Markovian approach. Int. J. Oper. Prod. Manag. **37**(1), 75–104 (2017). https://doi.org/10.1108/IJOPM-03-2015-0179.
74. Memon, A.M., Pollack, M.E., Soffa, M.L.: Automated test oracles for GUIs. In: ACM SIGSOFT Software Engineering Notes, vol. 25, pp. 30–39. ACM, New York (2000)
75. Microsoft Zune affected by 'bug' (2008). http://news.bbc.co.uk/2/hi/technology/7806683.stm
76. Muhammed, T., Mehmood, R., Albeshri, A., Katib, I.: Ubehealth: A personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities. IEEE Access **6**, 32,258–32,285 (2018). https://doi.org/10.1109/ACCESS.2018.2846609
77. Naish, L., Lee, H.J., Ramamohanarao, K.: A model for spectra-based software diagnosis. ACM Trans. Softw. Eng. Methodol. **20**(3), 11:1–11:32 (2011)
78. Nguyen, H.D.T., Qi, D., Roychoudhury, A., Chandra, S.: Semfix: Program repair via semantic analysis. In: Proceedings of the 2013 International Conference on Software Engineering, pp. 772–781. IEEE, Piscataway (2013)
79. Nidhra, S., Dondeti, J.: Black box and white box testing techniques-a literature review. Int. J. Embed. Syst. Appl. **2**(2), 29–50 (2012)
80. Offutt, J., Ma, Y.S., Kwon, Y.R.: The class-level mutants of MuJava. In: Proceedings of the 2006 International Workshop on Automation of Software Test, pp. 78–84. ACM, New York (2006)
81. Overview of the internet of things. Recommendations ITU-T Y.2060 (2012)
82. Perkins, J.H., Kim, S., Larsen, S., Amarasinghe, S., Bachrach, J., Carbin, M., Pacheco, C., Sherwood, F., Sidiroglou, S., Sullivan, G., Wong, W.F., Zibin, Y., Ernst, M.D., Rinard, M.: Automatically patching errors in deployed software. In: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, SOSP '09, pp. 87–102. ACM, New York (2009)
83. Pytlik, B., Renieris, M., Krishnamurthi, S., Reiss, S.P.: Automated fault localization using potential invariants. arXiv preprint cs/0310040 (2003)
84. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
85. Raghupathi, W., Raghupathi, V.: Big data analytics in healthcare: promise and potential. Health Inf. Sci. Syst. **2**(1), 3 (2014)
86. Renieres, M., Reiss, S.P.: Fault localization with nearest neighbor queries. In: Proceedings of the 18th IEEE International Conference on Automated Software Engineering, pp. 30–39. IEEE, Piscataway (2003)
87. Saraph, P., Kandel, A., Last, M.: Test set generation and reduction with artificial neural networks. In: Artificial Intelligence Methods in Software Testing, pp. 101–132. World Scientific, Singapore (2004)
88. Suma, S., Mehmood, R., Albugami, N., Katib, I., Albeshri, A.: Enabling next generation logistics and planning for smarter societies. Procedia Comput. Sci. **109**, 1122–1127 (2017). https://doi.org/10.1016/j.procs.2017.05.440. http://www.sciencedirect.com/science/article/pii/S1877050917311225. 8th International Conference on Ambient Systems, Networks and Technologies, ANT-2017 and the 7th International Conference on Sustainable Energy Information Technology, SEIT 2017, 16–19 May 2017, Madeira
89. Sun, Y., Han, J.: Mining heterogeneous information networks: a structural analysis approach. Acm SIGKDD Explor. Newsl. **14**(2), 20–28 (2013)
90. Visser, W., Păsăreanu, C.S., Khurshid, S.: Test input generation with Java pathfinder. ACM SIGSOFT Softw. Eng. Notes **29**(4), 97–107 (2004)

91. Wei, Y., Pei, Y., Furia, C.A., Silva, L.S., Buchholz, S., Meyer, B., Zeller, A.: Automated fixing of programs with contracts. In: Proceedings of the 19th International Symposium on Software Testing and Analysis, ISSTA '10, pp. 61–72. ACM, New York (2010)
92. Weimer, W., Nguyen, T., Le Goues, C., Forrest, S.: Automatically finding patches using genetic programming. In: Proceedings of the 31st International Conference on Software Engineering, ICSE '09, pp. 364–374. IEEE Computer Society, Washington, D.C. (2009)
93. Weimer, W., Forrest, S., Le Goues, C., Nguyen, T.: Automatic program repair with evolutionary computation. Commun. ACM **53**(5), 109–116 (2010)
94. Weyuker, E.J.: On testing non-testable programs. Comput. J. **25**(4), 465–470 (1982)
95. White, D.R., Arcuri, A., Clark, J.A.: Evolutionary improvement of programs. IEEE Trans. Evol. Comput. **15**(4), 515–538 (2011)
96. Zaidman, A., Van Rompaey, B., Demeyer, S., Van Deursen, A.: Mining software repositories to study co-evolution of production & test code. In: 1st International Conference on Software Testing, Verification, and Validation, pp. 220–229. IEEE, Piscataway (2008)
97. Zhou, Z.Q., Huang, D., Tse, T., Yang, Z., Huang, H., Chen, T.: Metamorphic testing and its applications. In: Proceedings of the 8th International Symposium on Future Software Technology (ISFST 2004), pp. 346–351 (2004)
98. Zikopoulos, P., Eaton, C., et al.: Understanding big data: analytics for enterprise class hadoop and streaming data. McGraw-Hill, New York (2011)

Chapter 22
Open Source and Open Data Licenses in the Smart Infrastructure Era: Review and License Selection Frameworks

Emad Alamoudi, Rashid Mehmood, Wajdi Aljudaibi, Aiiad Albeshri, and Syed Hamid Hasan

22.1 Introduction

According to the Open Source Initiative [1], open source licenses are *"licenses that comply with the* Open Source Definition*—in brief, they allow software to be freely used, modified, and shared."* The expression open source was first used by Bruce Perens and Eric Raymond in 1997 [2]. They wrote the Open Source Definition (OSD), which is based on ten criteria to determine whether a license is qualified to be an open source license [3].

The use of open source software (OSS) has increased over the years, particularly during the last two decades. In 2010, a study [4] showed that 98% of all enterprises use an open source software. Nevertheless, the use of open source is not restricted to businesses only; 76% of all software developers had acknowledged that they had used open source components in their software [4]. OSS give the user the right to use, change, and publish the software source code. However, there are some restrictions regarding its protection and copyrighting. These limitations are

E. Alamoudi (✉) · W. Aljudaibi · A. Albeshri
Department of Computer Science, Faculty of Computing and Information Technology (FCIT), King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: ealamoodi0004@stu.kau.edu.sa; waljedaibi@kau.edu.sa; aaalbeshri@kau.edu.sa

R. Mehmood
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: RMehmood@kau.edu.sa

S. H. Hasan
Department of Information Systems, Faculty of Computing and Information Technology (FCIT), King Abdulaziz University, Jeddah, KSA
e-mail: shhasan@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_22

presented in the software license. The license serves as a contract between the developer of the software and the end user who will use the software.

The open source software culture has helped the development of many new distributed and collaborative applications paving the way for integrated systems and hence smart cities. Many new smart city applications are being developed, such as in transport [5–12], healthcare [13–16], infrastructure [17, 18], and applications [19, 20].

The open data licenses have received an increased attention in the recent years due to the emergence of big data and relevant technologies. Big data refers to the *"emerging technologies that are designed to extract value from data having four Vs characteristics; volume, variety, velocity and veracity"* [21]. Big data technologies are being used in many application areas, see e.g., [9, 22–26].

Smart city developments are increasingly relying on data sharing concepts. Data sharing and "open data" are opening the doors for the use of timely data in making new products and services, and hence are allowing businesses, governments, and individuals to develop, accelerate, and innovate knowledge, science, technology, and economy. For example, Transport for London (TfL) has published the traffic information for London city under an open license to be used by developers [27]. TfL uses an Open Government License (OGL) v2.0 [28]. A study done in 2017 by Deloitte shows that opening up the data by TfL has generated about £130 million for the economy [29]. See e.g., [30–34], for other initiatives and success stories of data sharing.

Data and software licensing are playing an important role in the data sharing economy era. Making the data or software publicly available is not a simple matter due to privacy, competing interests of the parties involved, and many other reasons. The legal information of who could access and use the data and software, and how to use them, needs to be explicitly stated. These matters are dealt with by specifying data and software licenses. This has given rise to many types of licenses.

The license selection process is critical because a license, once picked, could hardly be changed in the future [35]. Very few cases had been made in the past to change licenses and had faced a furious resistance from contributors. The process requires a painful effort from licensors to acquire permission from each and every contributor [35]. More importantly, selecting a wrong license could have severe financial and social implications, cause risks, and hinder developments (see e.g., [36–38]).

This chapter proposes frameworks for the selection of open source software and open data licenses. A review of notable open source and open data licenses, their differences, and the suitability of these licenses for various kinds of data and software is carried out. Conclusions are drawn with recommendations for the future work.

The chapter is structured as follows. In Sect. 22.2, we give a brief overview of the open source license domain. Section 22.3 reviews the related works. Section 22.4 describes some notable open source applications in the smart city domain. In Sect. 22.5, we give an overview of the main categories of the open source software license with a brief explanation of each type. In Sect. 22.6, we review open data licenses and

briefly describe its main categories. In Sect. 22.7, we discuss the licenses which we have selected to be used in our frameworks and give justifications for their selection. Sections 22.8 and 22.9 present and discuss the software license and data license frameworks, respectively. Section 22.10 concludes the chapter.

22.2 Background

Why a developer gets involved in OSS projects? According to Kolassa and Rumpe [39], there are several reasons, such as showing their skills, showing the world their work, for altruistic reasons, and/or for possible rewards in the future.

There are two well-known organizations and movements that certify the OSS license: The OSI (Open Source Initiative) and the FSF (Free Software Foundation). The OSI, a non-profit corporation, was founded in 1998 for the purpose of reviewing and approving licenses as OSD-conformant [40], whereas the FSF, a non-profit organization, was established in 1984 by Richard Stallman [2]. It introduces one of the most notable licenses which is GNU General Public License (GPL) along with the concept of copyleft.

OSI had approved until this point 83 licenses [41] while FSF lists 98 open source software licenses [42]. Many licenses had been ratified in the past years. Also, a single license may have many versions, and each released might be distinct from its earlier versions. As a result, there is a growing need to simplify the disparity between the different types of licenses.

The open source concept can also be applied to hardware. Open Source Hardware Association (OSHWA) defines open source hardware as a hardware that has an available design to the public [43]. Same as open source software, open source hardware enables users to change, produce, and distribute the devices based on that design.

The "smart city," on the other hand, starts to get more attention recently. Different cities have already taken the start on the smart city initiatives [44]. Smart cities *"provide the state of the art approaches for urbanization, having evolved from the developments carried out under the umbrella of knowledge-based economy, and subsequently under the notion of digital economy and intelligent economy"* [25]. Caragliu et al. [45] *"believe a city to be smart when investments in human and social capital and traditional (transport) and modern (ICT) communication infrastructure fuel sustainable economic growth and a high quality of life, with a wise management of natural resources, through participatory governance."*

Tsarchopoulos et al. [46] argued that a smart software that enables reusability will facilitate the strategy of the smart city. Thus, software applications are an essential part of building smart cities. To develop an application that targets smart city problem, organization and city authorities should avoid spending extra money and technology lock-in. Therefore, open source paradigm will be a good strategy to adopt since it allows sharing application between cities, build collaboration, provide an application for free to other cities, and reuse existing application created by

others. Moreover, Komninos et al. [44] believe that open source software are perfect for city authorities because they will not be able to compete in software and do not create advantages over proprietary software.

Selecting an open license might be problematic. One of the challenges on the selecting process is when a developer uses an OSS component that is licensed under a particular OSS license. So, the final product license should be compatible with the component license. This problem is called license-mismatch [47]. As the number of components would increase, as the complexity of selecting which license to apply to the final software will be. For example, GPL version 3 is not compatible with components that are published under GPL version 2.

A solution for the license-mismatch problem, rather than combining the two components under an incompatible license, would be to form a new license which could include both different licenses restrictions. This solution is called license proliferation, and it is strongly discouraged by the OSI because it raises another type of incompatibilities [48]. Lately, different companies such as Palamida (palamida.com), BlackDuckSoftware (www.blackducksoftware.com), and Open-Logic (www.openlogic.com) offer to help clients by analyzing the possible legal outcomes when planning to use OSS components in their products. Another solution is to apply the concept of dual licensing [49]. However, this solution may rise further problems.

The licensor of the OSS software may be represented by an organization, a single developer, or a group of developers. On the other hand, the licensee could be the end user of the software or somebody who has embedded it in his application.

22.3 Related Work

Many publications addressed the topic of choosing the suitable license for a particular project from a different perspective.

Lindman et al. [50] propose an OSS license decision-making model which links the license with the business model. The paper also discusses various factors that should be taken into consideration when selecting a license. However, the suggested model was targeting a small size company. For the open data selection, the European data portal had a guide that can help in selecting the right data license [51]. Androutsellis-Theotokis et al. [2] provide a general overview of all aspects related to OSS such as history and evolution, licensing, reuse and adoption, communities, creation process, business models, and motivation. Furthermore, Laurent [52] went in-depth speaking about the most popular license in details. He addressed the advantage and disadvantage when using each one. Mathur et al. [47] analyze 1423 projects from Google Code project hosting that contain around 69 million lines of code. Their goal was to spot license violation by tracking cases of OSS code reuse. As a result, they found four types of violations. They provided two solutions to avoid similar violations. Kolassa and Rumpe [39] study the legal impact that a code generator's license can impose on the generated code or artifacts. Kechagia et al.

[53] discuss various features of OSS licensing. Also, they present a study based on FreeBSD port collection, which is a group of more than 20,000 software packages. They investigate their licensing dependency to find any sort of pattern. Their aim was to guide and explain the process of selecting an OSS license. Fitzgerald [54] identifies the characteristics of the original free and open source software (FOSS) phenomenon. Then, he tried to determine the new paradigm which he called OSS 2.0. He illustrates the key challenges for research and practice that occur as a result of the appearance of OSS 2.0. Later, he stated the differences of licensing selection process between the two phenomena. Kapitsaki et al. [55] investigate different tools that detect the license of a software component in order to avoid license violations. Then, authors propose their graph approach in identifying license compatibility. Reincke et al. [56] offer an easy and reliable tool to fulfill what one has to do in order to use open source compliantly. Their tools have been authorized by OSI as a tool that one may use to manage the open source compliance. It provides a to-do list that must be executed to ensure that user act in accordance to open source license requirements. Singh and Phelps [35] present a study of social factors that influence the choice of open source license. They test their hypotheses on a sample consisting of 5307 open source projects hosted by SourceForge. Heikkilä et al. [57] discuss the problem of transferring process from proprietary development model to open source model. Furthermore, they demonstrate challenges and benefits that might be expected from such a process and how license choice could affect this transition. Widenius and Nyman [58] try to answer the question of how can someone make money out of open source project? They compare common OSS license then they identify one that suits their purpose. Tsarchopoulos et al. [46] has created a special repository for open and proprietary application, which they call ICOS (Intelligent City Software and Solutions). Their repository provides a platform for uploading and sharing intelligent and smart city application. They also provide a forum for discussion between users. Amiri-Kordestani and Bourdoucen [59] review some state-of-art open source applications used in the field of IoT. They also listed a number of parameters that need to be studied before adopting open source project.

The challenges of 2014 mission of GBIF (Global Biodiversity Information Facility) to provide a machine readable, standard license for American bullfrog records downloaded from GBIF are discussed in [60, 61].

Most of the papers and books that previously discussed were dealing with the problem from the business point of view. However, in our paper, we will propose a simple yet comprehensive model that helps to pick the right license for developers who are not interested to gain profit out of their open source product.

22.4 Open Source Software, Projects, and Applications

As smart cities model advance, lots of software are being developed to compensate for its rising needs. Existing large-scale software repositories, namely, GitHub, Bitbucket, and SourceForge can be used to host smart city applications. However,

Table 22.1 Open source projects for IoT with their associate licenses

Application	License	Purpose
IoTivity [59, 62]	Apache License version 2.0.	An open source framework of the connectivity standards for IoT devices
Tizen [59, 63]	Tizen SDK License, Flora License	An operating system for many embedded devices, such as smartphones, tablets, TVs, cameras, printers, wearables, and home appliances
Automotive Grade Linux [59, 64]	Apache License 2.0	It is an embedded Linux project to create open source software with members from the silicon industries, automotive, and telecom
Yocto [59, 65]	MIT License	It is an embedded Linux project that offers a framework to make a highly customized Linux distribution
Zephyr [59]	Apache 2.0	An embedded and real-time operating system (RTOS) targeting microcontrollers
Android Things [59]	Apache 2.0/GPLv2	It is built to assist high-end IoT devices, wireless networking, and sensors

special repositories for only smart city applications can be found, such as Apps for Barcelona, the Code for America, and ICOS [46]. Such a repository will enable sharing and reusing smart city application, which indicates some degree of maturity of smart city development. ICOS currently holds 83 projects classified by their own function and software license. Out of the 83 projects, 69 (83%) have open source licenses, which shows the popularity of open source software in the smart city applications. Table 22.1 shows some examples of open source projects in the smart city domain.

22.5 Open Source Software Licenses

Choosing a license is a very critical task because each license could flourish or shrink the community around your software.

All open source licenses assure the user the right to use, modify, and redistribute the source code in its original or modified form. Rather than those fundamental rights, licenses may differ considerably. Open source software licenses are usually divided based on how they treat derivative works into three categories: (1) Permissive, (2) Weak copyleft, and (3) Strong copyleft licenses.

A comparison between the three categories is presented in Table 22.2.

A number of open source licenses that are approved by OSI divided into the three categories are presented in Table 22.3.

The hierarchical structure is presented in Fig. 22.1 along with some examples for each category.

Table 22.2 Open source license categories

	Permissive	Weak copyleft	Strong copyleft
Free of cost	√	√	√
Redistribution	√	√	√
No usage restrictions	√	√	√
The availability of the source code	√	√	√
Source code modification	√	√	√
Integrating with proprietary work	√	√	×
Derivative work can be proprietary	√	×	×
Can be relicensed by others	×	×	×

Based on information from [53]

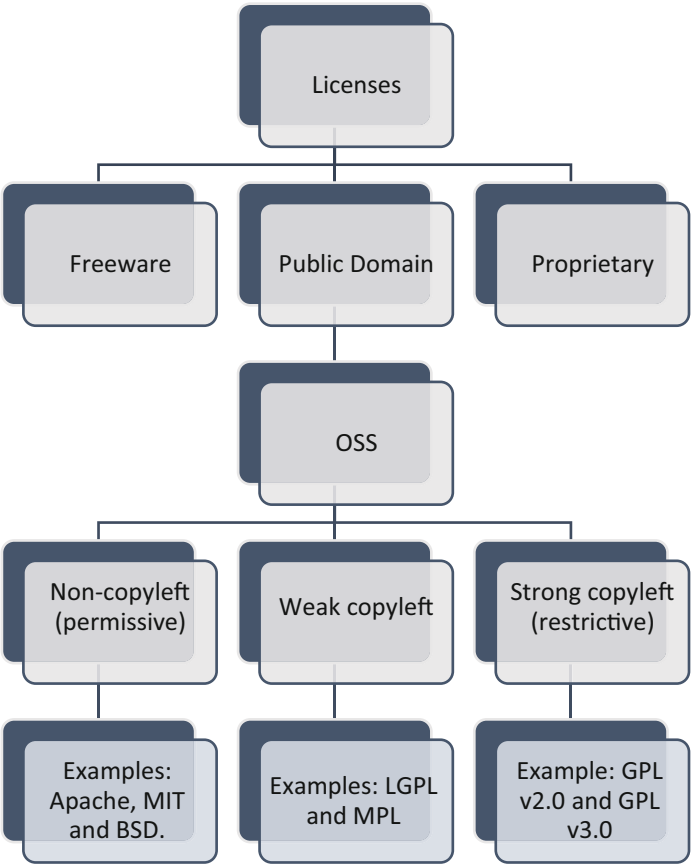


Fig. 22.1 A hierarchical structure of software license categories

Table 22.3 Some OSI-approved licenses divided into three groups

Permissive	Weak copyleft	Strong copyleft
Apache License v. 2.0	Adaptive Public License v. 1.0	Frameworx License (Frameworx-1.0)
Boost Software License (BSL-1.0)	Artistic License (Perl)	GNU Affero General Public License v. 3 (AGPL-3.0)
BSD 3-Clause "New" or "Revised"	Apple Public Source License (APSL-2.0)	GNU General Public License v. 2.0 (GPL-2.0)
BSD 3-Clause "Simplified" or "FreeBSD"	Eclipse Public License (EPL)	GNU General Public License v. 3.0 (GPL-3.0)
Academic Free License ("AFL") v. 3.0	GNU Lesser General Public License v. 2.1 (LGPL-2.1)	Non-Profit Open Software License v3.0 (NPOSL-3.0)
Attribution Assurance Licenses (AAL)	GNU Lesser General Public License v. 3.0 (LGPL-3.0)	Open Software License v. 3.0 (OSL-3.0)
EU DataGrid Software License (EUDatagrid)	Microsoft Reciprocal License (Ms-RL)	Reciprocal Public License v. 1.5 (RPL-1.5)
Educational Community License, v. 2.0 (ECL-2.0)	Motosoto License (Motosoto)	Sleepycat License (Sleepycat)
Eiffel Forum License v. 2.0 (EFL-2.0)	Mozilla Public License v. 2.0 (MPL-2.0)	
Entessa Public License v. 1.0	Nokia Open Source License (Nokia)	
Fair License	Ricoh Source Code Public License (RSCPL)	
ISC License (ISC)	Sun Public License v. 1.0 (SPL-1.0)	
MirOS Licence	wxWindows Library License (WXwindows)	
MIT License (MIT)		
PHP License v. 3.0 (PHP-3.0)		
PostgreSQL License (PostgreSQL)		
Python License (Python-2.0)		
CNRI Python License (CNRI-Python)		
Vovida Software License v. 1.0 (VSL-1.0)		
W3C License (W3C)		
X.Net License (Xnet)		
Zope Public License v. 2.0 (ZPL-2.0)		

Based on material from [42, 55]

22.5.1 Permissive Licenses

Permissive licenses are the most liberal category. In contrast to weak copyright licenses, there are no restrictions to give back the modifications to the community. It gives the user the right to use and reuse (or even fork) the source code. It only requires licensees to give credit to the original contributors. By using this type, people could fork the source code and release it under a proprietary license. An example of a forking case was Apple's OS X operating system which contains code that copied from the FreeBSD operating system, which released under BSD (Berkeley Software Distribution) license. Sometimes people referred to this type as "academic licenses." The most common examples of this category are the Apache, MIT, and BSD licenses.

22.5.2 Weak Copyleft Licenses

Weak copyleft license enables users to combine their software with proprietary software. It, hence, establishes a middle layer between the strong copyleft licenses that prohibit such merging, and the permissive licenses that allow this without any control. However, this type cannot be relicensed under a proprietary license. Any modification to code under this category should be made available as an open source too. Some examples of weak copyleft licenses are the Mozilla Public License (MPL), which is the license used with the Firefox internet browser and GNU Lesser General Public License (LGPL), which is used with the Linux OS libraries [62].

22.5.3 Strong Copyleft Licenses

An essential feature of the strong copyleft licenses, which lead to its extensive approval, is the viral nature, which requires that any modification to the source code to also be released under a strong copyleft license [53]. Programs that have been released under this type of license cannot be relicensed under a more permissive license. The most notable example is GNU General Public License (GPL), which has been used for Linux kernel [62].

A general rule about changing a license is that user can switch to a license that is more restrictive license type yet not to a more permissive one. Moreover, only the permissive license type can be modified to proprietary [58].

22.6 Open Data License

In a smart city context, data is expected to be generated massively. Sensors, cameras, smartphones, among others, all are expected to generate information. However, without opening these data, developers will not be able to fully utilize the data. Open data can greatly help speed up innovation and invention, which is going to help opening new businesses and jobs. Yet, open data is a term that can only be given to the data that has an open license [51]. Open data license can help in regulating the legal conditions about how to use, distribute, and modify these data. Different countries are competing to provide their data. Global Open Data Index [63] ranks countries on how much data they offer. Table 22.4 shows the open data portal for a number of countries along with their ranking.

Open data licenses can be divided into three categories according to the restrictions that they put on the users. First, the public domain category, which waives all the author rights and puts no restrictions on the user. Second, the attribution category, which gives the user the freedom to use, modify, and share the data as long as the user give credit to the original author. Finally, a share-alike category which requires that all derivative work to be shared (Fig. 22.2).

Table 22.4 The data portal websites for a number of countries

Country	Open data website	Rank
Taiwan	data.gov.tw	1
Australia	data.gov.au	2
United Kingdom	data.gov.uk	3
France	data.gouv.fr	4
India	data.gov.in	32
Saudi Arabia	data.gov.sa	–

Ranking data were collected from: https://index.okfn.org/place/

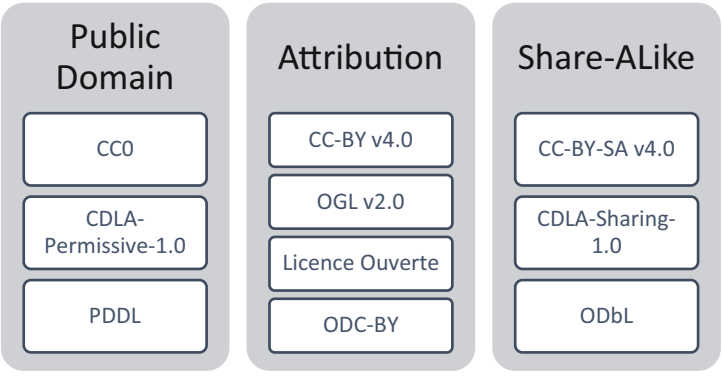


Fig. 22.2 Open data licenses divided into different categories according to the restrictions that they place on the user

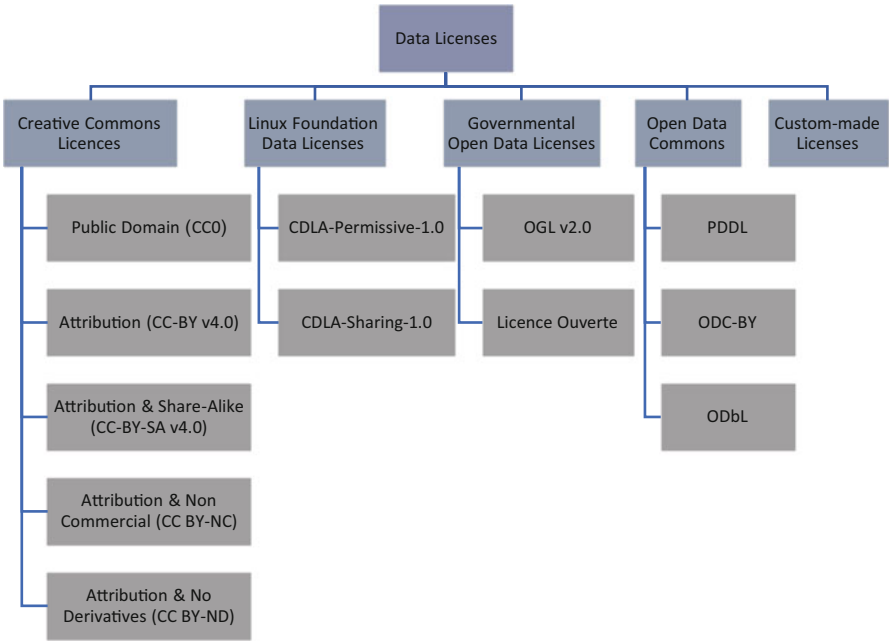


Fig. 22.3 A hierarchical structure of data license categories according to their issuers

Moreover, one can also divide them according to the source that issues the licenses. There are five major issuers of open data licenses, namely, creative commons licenses, Linux Foundation, governmental open data licenses, open data commons, and custom-made licenses [27] (Fig. 22.3).

Creative Commons Licenses First, the Creative Commons (CC) licenses are a group of licenses that used for open content. It is used to give the user the right to use, distribute, and build upon a work that is already created. Version 4.0 explicitly considers licensing data. Yet, the CC licenses, in case of data, can be further divided into three licenses, namely, public domain, attribution, and share-alike.

Linux Foundation Data Licenses Linux foundation has several projects, one of which is called Community Data License Agreement (CDLA). CDLA is a community work to develop data licenses similar to the one that was created for software which enables using and sharing the data. CDLA collaboration resulted in two open data licenses, CDLA-Permissive v1.0 and CDLA-Sharing v1.0.

Governmental Open Data Licenses Governments usually own a large volume of data. These data might be tax, power consumption, water usage, crime reports, and business data. Thus, these data are routinely collected and being ready to be navigated and mined. For maximum utilization, governments start to open these data for the public. In order to release the data, governments usually issue data license to regulate the legal issues. Governments usually publish their data by using their

own license. Open Government License (OGL), which is published by the United Kingdom government, is an example of a governmental license that respects the open data terms [64]. OGL gives the user the right to use, share, copy, and distribute the data. It also allows exploiting the data commercially, and it has two versions (v1.0 and v2.0). However, it requires to mention the source of the data by providing an attribution statement [64]. OGL is compatible with both the Creative Commons Attribution License v4.0 (CC-BY v4.0) and the Open Data Commons Attribution License (ODC-BY) [64].

Another example of a clear and open data license is the License Ouverte, which was issued in 2012 by the French government. Similar to the OGL, the license ouverte allows the user to use, share, modify, and commercial exploitation of the data while requiring the user to attribute the source of the data [65]. It is compatible with other licenses like OGL, ODC-BY, and CC-BY 2.0.

Open Data Commons Licenses Open Data Commons (ODC) project provides legal solutions for open data. It had launched its first license in 2008. ODC is a project that was carried out by the Open Knowledge Foundation, which is a global non-profit organization [66]. ODC has three open data licenses, namely, PDDL (Public Domain Dedication and License), ODC-BY (Attribution License), and ODC-ODbL (Open Database License). PDDL license waives all author rights [67]. There are no obligations on the user when using PDDL. ODC-BY is similar to PDDL, but it requests an attribution. Finally, ODC-ODbL license enables the user to use, share, and distribute the data, but it asks for attribution and to license all derivative works under the same license [68].

Custom-made licenses Rarely, a data publisher will share his data under a specific custom-made license. The author might need to do that in case that current licenses do not cover his specifications. Custom-made licenses can be written by the data owner or adapted from an existing standard license through the insertion of new specifications and/or the adjustment of existing licenses.

However, usually custom-made licenses increase the legal complexity for the end user. Moreover, the extra specification might violate the open data standards or limit the freedom of the user [27].

22.7 Selected Licenses

22.7.1 Software Licenses

In our framework for selecting open source software license, we will only include the most notable licenses. The license popularity will be taken from the black duck license usage statistics [69]. These statistics are calculated from the Black Duck KnowledgeBase which includes 1.1 million open source projects from over than 8500 sites.

Table 22.5 List of licenses and their approval status

License name	Type	FSF approval	OSI approval
MIT	Permissive	√	√
Apache License 2.0	Permissive	√	√
GNU General Public License (GPL) 3.0	Strong Copyleft	√	√
BSD License 2.0 (3-clause, New or Revised)	Permissive	√	√
Artistic License (Perl)	Weak Copyleft	√	√
GNU Lesser General Public License (LGPL) 3.0	Weak Copyleft	√	√
Eclipse Public License (EPL)	Weak Copyleft	√	√
GNU Affero General Public License v3 (AGPL)	Strong Copyleft	√	√
The GNU Free Documentation License (GFDL)	–	√	×

Based on material from [41, 42, 70]

We looked for the ten most broadly used licenses and then decided to eliminate older versions of each license. Moreover, we removed the Microsoft Public License since it is similar to the Eclipse Public License (EPL) which is more valuable in the context of model-driven development. We end up having seven licenses for our analysis. Furthermore, we add the GNU Free Documentation License (GFDL) and GNU Affero General Public License (AGPL) since they tackle areas that are not touched by the other licenses. Table 22.5 shows all the covered licenses along with their types and the acceptance status.

MIT MIT license is another permissive license. It is considered to be the simplest on its category [52]. The license was written in 1987 for the purpose of licensing the source code of the X Window System.

Apache License 2.0 Apache license was created by the Apache Foundation. It has been derived from BSD license. Version 2.0 had been written in 2004. It shares some similarities with both MIT and BSD licenses. However, licensing the modifications under another license is allowed as long as the requirements of the Apache License v2.0 are complied with (This has been implying in BSD and MIT but not spelled out) [2]. It allows contributors to grant their patent right from users [71].

GNU General Public License (GPL) 3.0 GPL is one of the essential open source licenses. It was created by Richard Stallman in the mid of 1980s. GPL enforces any derivative works or modifications to be published under the same GPL licenses. Hence, it assures to its users that their work would never convert to a proprietary product [72]. GPL is the heart of the concept of copyleft. Most OSS to date are released under GPL. According to Black Duck KnowledgeBase, 30% of all OSS project are released under different versions of GPL [69]. GPL v3.0 released in 2007 included patent protection clauses.

Examples of software that use GPL are C Compiler, Linux operating system kernel, and the GNU Emacs Editor.

BSD License 2.0 (3-clause, New or Revised) License The Berkeley Software Distribution (BSD) license was created for the purpose of licensing a big portion of the Unix operating system code by the University of California (UC) [2]. Since then, BSD license was gradually being adopted by many OSS. It permits derivative works to be licensed under different licenses, as long as it gives credit to the original contributors. However, the clause that requires the acknowledgment of all previous contributors dropped in 1999.

Artistic License (Perl) Artistic License (AL) was introduced firstly in 1991 by Larry Wall for Perl. The reason behind creating AL was that Larry felt that conditions of GPL (which used to be Perl's license) was very restricted, so he was looking for a much flexible license. The aim of this license was to permit Perl programs to be integrated into commercial packages [2].

Its name reflects the author's intention to preserve his artistic control over the licensed product and its derivative works [52]. It enables developers to do anything with the source code if only the changes they made are declared and explained in the source code. Thus, it allows the original author to keep his artistic control [72].

The AL and GPL are pretty similar licenses. However, AL is a weak copyleft license because it doesn't require its derivative works to be published using the same license.

GNU Lesser General Public License (LGPL) 3.0 The GNU Lesser General Public License (LGPL), which also known as the Library GPL, was introduced by FSF as a tactic to protect the free software library against more permissive licenses. Therefore, this tactic allows the library to be combined with proprietary software. Thus, the library would have bigger chance to be widely adopted. The main difference between LGPL and GPL is that program and library released under LGPL could be incorporated into a proprietary software, or one that is licensed under non-LGPL license. It is referred to as a weak copyleft license.

However, the FSF advocates the use of GPL license in cases when there is a library that provides unique and extraordinary advantages that are not found in similar competing libraries [2].

Eclipse Public License (EPL) Eclipse public license was initially driven from IBM's Common Public License (CPL), which was used in the past for licensing the Eclipse codebase. The only difference between EPL and CPL is related to how they deal with the patent litigation. The EPL was created for the Eclipse Foundation. There are two versions which are ELP v1.0. and ELP v2.0. Both versions have been approved by the OSI as a qualified OSS license that satisfied the OSD requirements on May 2004 and August 2017, respectively [73, 74]. Furthermore, both are considered as weak copyleft licenses by the FSF [42]. The only difference that ELP v2.0 offers is that it allows the GNU GPL version 2 or later as a "secondary license" [42].

GNU Affero General Public License v3 (AGPL) The GNU Affero General Public License v3 (AGPL) is a copyleft license that was created by the FSF. It is similar to GPL v3 with one additional term that allows users who deal with an AGPL-licensed software over a network to acquire the source code of that software [42]. Hence, this license is recommended by FSF for any software that runs over the network. However, the license is not compatible with both GPL v2 and v3.

The GNU Free Documentation License (GFDL) GFDL was created for licensing manuals, textbooks, or other documents. It allows everyone to copy, modify, and redistribute the material freely either commercially or non-commercially.

22.7.2 Open Data Licenses

In this section, we will consider data licenses from trusted sources and organizations. There are four major open data license providers: creative commons licenses, Linux Foundation, governmental open data licenses, and open data commons. From each one, we will select the most notable ones.

From the creative commons licenses, we will select the CC0, CC-BY v4.0, and CC-BY-SA v4.0. Furthermore, we will pick the shared and permissive licenses from the Linux foundation. Finally, three licenses will be selected from the open data commons, which are PDDL, and ODC-BY.

CC0 License CC0 is one of the creative commons licenses. It waives all the legal restriction on the data. So, the user can use the data without any obligation.

CC-BY License v4.0 CC-BY v4.0 allows the user of the data to use, change, and distribute the data as long as he gives the attribution to the original author.

CC-BY-SA License v4.0 CC-BY-SA v4.0 gives the user the right to use, modify, and share the data as long as the original source of the data is attributed. Moreover, it requires that any derivative work to be published under the same license.

CDLA-Permissive v1.0 License CDLA-Permissive v1.0 allows the user of the data to use, share, and modify the data without any legal obligations [75]. No need to share any derivative works when using the CDLA-Permissive.

CDLA-Sharing v1.0 License The CDLA-Sharing v1.0 license allows the user to use and modify the data under the license; yet, the user is required to publish any changes and modifications that he made [76].

PDDL License PPDL license is one of the results of the open data common project. It gives the user to use the data in any form without any legal restriction.

ODC-BY License Similar to PPDL, ODC-BY license allows the user to use, edit, and distribute the data. However, ODC-BY asks for attribution to the original author.

22.8 The Open Source Software License Selection Framework

Different factors could affect the decision of picking the license to an OSS software. However, there are some general rules better to keep in mind when choosing a license. First, try to select an existing license that is empirically worked, rather than creating a new one [72, 77]. Using a trusted and reputable license can give confidence toward your software. On the other hand, using an unpopular license might create ambiguity and confusion. In addition, to create a license from scratch, you need to have a solid legal background.

Nonetheless, the selecting process sometimes may be restricted by pre-existing applications used in the software. For instance, if GPL licensed software is used, derivative works would have no choice but to use GPL.

Here, we will illustrate a useful yet straightforward framework to pick the right license for your software. The framework is presented in Fig. 22.4 with all different cases.

When deciding which license to select for your software, one of the following scenarios might occur:

Case 1. You need to be sure that there is no pre-existing license on the software or at least there is no strong copyleft one. A strong copyleft license would limit the developer options to stick with the same old license. However, permissive type would allow the user to pick any license for the derivative work.

Case 2. In the case if the developer wants his code to stay open source yet be able to be combined with a proprietary license, then EPL would be a good choice. EPL enables an open source software to be combined with non-open source software without any collision.

Case 3. Then, a developer should ask himself whether he cares about sharing the improvements done to his software. If yes, then he should consider a strong copyleft license. We would recommend user to select GPL 3.0 since it vigorously enforces derivative works to stick with the same license. If a user selects GPL to his software, he will guarantee that any modification would be released under the same license. So, the software (and its derivative works) would always be strong copylefted and immune from forking.

Case 4. If the developer is not concern about the chance that people might fork his work, yet he wants to maintain access to the original version and modification. Then, we advise him to use AL license. AL allows the original author to maintain his artistic control over the source code.

Case 5. However, if the developer is concern about using a very liberal and simple license that gives the user the right to do anything with the source code, then we would recommend using either MIT or BSD License v2.0. Both come under the permissive category, and they only require the user to give attribution back to the original author.

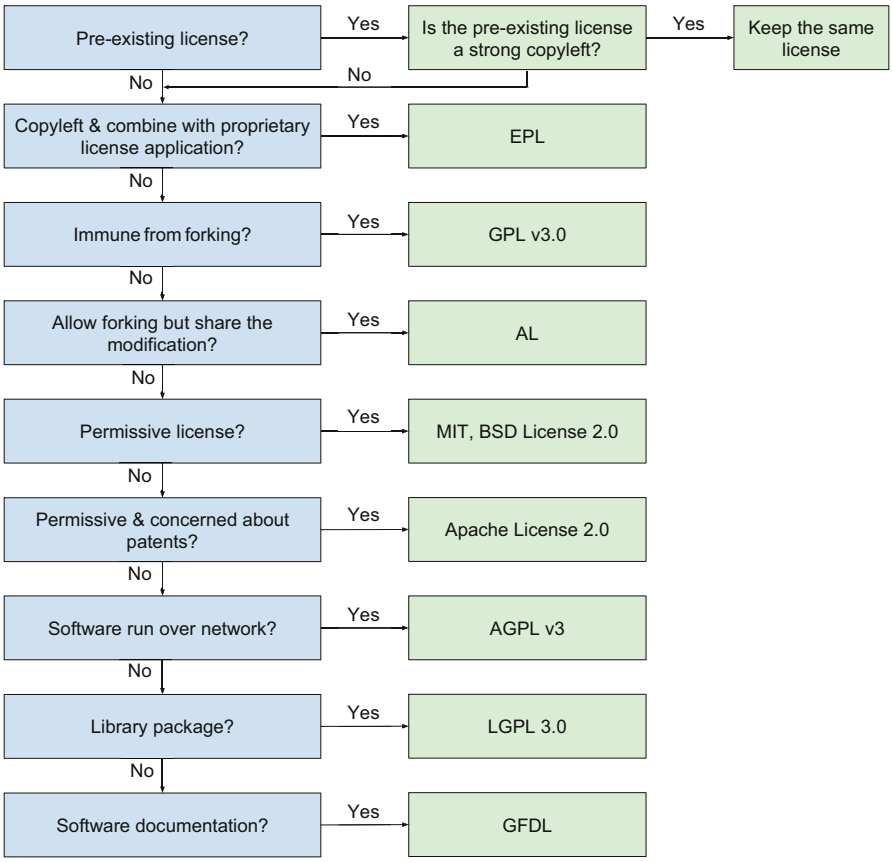


Fig. 22.4 A flowchart of the open source software license selection framework

Case 6. When developer seeks to publish his software under a permissive license but still he wants to retain his patent rights from users, then Apache License 2.0 is an advised pick. It allows contributors to grant their patent rights.

Case 7. If a developer wants a license that is similar to GPL while it is still compatible with web applications, then AGPL would be the solution. It retains the same restrictions as the GPL but with an extra sentence which required that the web publication should be distributed beside the source code.

Case 8. For libraries, LGPL v3 is a suitable option. It requires that library's derivative works to be released under the same LGPL, but applications that utilize that library do not have to stick with the same license.

Case 9. Finally, when a developer wants to license manual, textbook, or other functional and useful documents, GFDL would be recommended. It gives everybody the right to use, redistribute, modify, and sell the documentation. It gives credit to the original creator while eliminating any responsibility for modifications done by others.

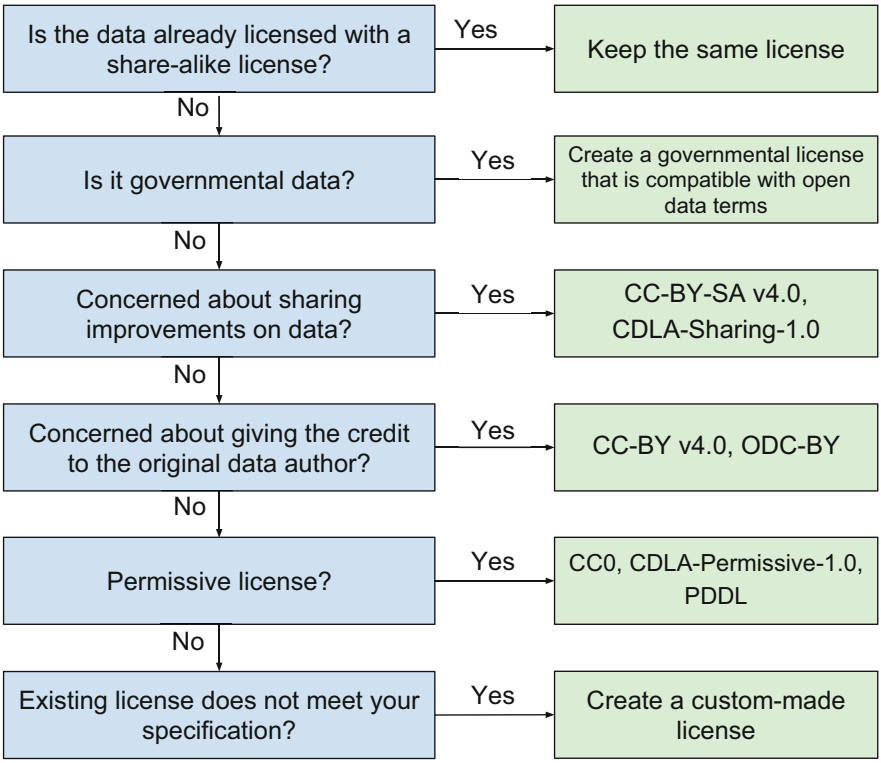


Fig. 22.5 A flowchart of the open data licenses selection framework

22.9 The Open Data License Selection Framework

Similar to what we have done in the software licenses, we created a framework that helps to pick the right open data license. The framework (see Fig. 22.5) consists of six cases:

Case 1. If the data that is being licensed is already licensed under the share-alike category, then it is better to keep the same license.

Case 2. If a government authority wants to publish a governmental open data, then it is better to create an open data license that is compatible with the open data standards. The license is better to be short, compatible with notable existing licenses, and easy to follow.

Case 3. If the author wants to keep all derivative work out of his data to be published and shared, then share-alike category can help to achieve this goal. CC-BY-SA v4.0 and CDLA-Sharing v1.0 licenses are the ones to consider in such a scenario.

Case 4. If the data owner wants to share his data, and at the same time he wants to get the credit for his work, then he should use attribution licenses. Two suggested licenses under these categories are: ODC-BY and CC-BY v4.0.

Case 5. In case that the owner does not want to put any restrictions upon data, then we would suggest public domain license category. In this situation, we would recommend three licenses: CC0, CDLA-Permissive-1.0, or PDDL, each of which serves the same purpose.

Case 6. Finally, if all previous licenses do not cover all the specifications of the data owner, then he can create his own open data license. This step is not recommended since it is going to create more complexity for users of open data.

22.10 Conclusion

Open source software has remarkably increased lately paving the way for many innovations such as Internet of Things (IoT) and smart cities. The introduction of technologies like big data has given rise to open data licenses, and the "Share more—Develop less" culture, which in turn have raised new legal challenges. The community has been developing many new licenses to address these emerging legal issues. However, selecting the right license is becoming more difficult due to the licensing complexities and continuous arrival of new licenses. The license selection process is critical because selecting a wrong license might lead to serious financial and social consequences.

In this chapter, we gave a brief overview of the open source license domain with a review of related works. Furthermore, we described some notable open source applications in the smart city domain. Then, we gave an overview of the main categories of the open source software license with a brief explanation of each type. We reviewed open data licenses and briefly described its main categories. Finally, we presented the software license and data license frameworks, respectively.

Future work will focus on improving the quality of the proposed frameworks by including more licenses and making the selection process more robust.

References

1. Licenses & Standards | Open Source Initiative. https://opensource.org/licenses. Last accessed 30 July 2018
2. Androutsellis-Theotokis, S., Spinellis, D., Kechagia, M., Gousios, G.: Open Source Software: a survey from 10,000 feet. Found. Trends® Technol. Inf. Oper. Manag. **4**, 187–347 (2010)
3. Perens, B., Sroka, M.: The Open Source Definition (2007)
4. Zenoss Releases 2010 Open Source Systems Management Survey Report. http://www.marketwired.com/press-release/zenoss-releases-2010-open-source-systems-management-survey-report-1302973.htm. Last accessed 30 July 2018.

5. Aqib, M., Mehmood, R., Albeshri, A., Alzahrani, A.: Disaster management in smart cities by forecasting traffic plan using deep learning and GPUs. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, pp. 139–154. Springer, Cham (2018)
6. Alam, F., Mehmood, R., Katib, I.: D2TFRS: an object recognition method for autonomous vehicles based on RGB and spatial values of pixels. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, pp. 155–168. Springer, Cham (2018)
7. Muhammed, T., Mehmood, R., Albeshri, A.: Enabling reliable and resilient IoT based smart city applications. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, pp. 169–184. Springer, Cham (2018)
8. Al-Dhubhani, R., Mehmood, R., Katib, I., Algarni, A.: Location privacy in smart cities era. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, pp. 123–138. Springer, Cham (2018)
9. Suma, S., Mehmood, R., Albeshri, A.: Automatic event detection in smart cities using big data analytics. In: International Conference on Smart Cities, Infrastructure, Technologies and Applications SCITA 2017: Smart Societies, Infrastructure, Technologies and Applications, pp. 111–122. Springer, Cham (2018)
10. Alomari, E., Mehmood, R.: Analysis of tweets in Arabic language for detection of road traffic conditions. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, pp. 98–110. Springer, Cham (2018)
11. Arfat, Y., Aqib, M., Mehmood, R., Albeshri, A., Katib, I., Albogami, N., Alzahrani, A.: Enabling smarter societies through mobile big data fogs and clouds. Procedia Comput. Sci. **109**, 1128–1133 (2017)
12. Schlingensiepen, J., Nemtanu, F., Mehmood, R., McCluskey, L.: Autonomic transport management systems—enabler for smart cities, personalized medicine, participation and industry grid/industry 4.0. In: Intelligent Transportation Systems–Problems and Perspectives, pp. 3–35. Springer, Cham (2016)
13. Alotaibi, S., Mehmood, R.: Big data enabled healthcare supply chain management: opportunities and challenges. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, pp. 207–215. Springer, Cham (2018)
14. Alamoudi, E., Mehmood, R., Albeshri, A., Gojobori, T.: DNA profiling methods and tools: a review. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, pp. 216–231. Springer, Cham (2018)
15. Al Shehri, W., Mehmood, R., Alayyaf, H.: A smart pain management system using big data computing. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 232–246. Springer, Cham (2018)
16. Khanum, A., Alvi, A., Mehmood, R.: Towards a semantically enriched computational intelligence (SECI) framework for smart farming. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, pp. 247–257. Springer, Cham (2018)
17. Usman, S., Mehmood, R., Katib, I.: Big data and HPC convergence: The cutting edge and outlook. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, pp. 11–26. Springer, Cham (2018)
18. Ahmed, W., Khan, M., Khan, A.A., Mehmood, R., Algarni, A., Albeshri, A., Katib, I.: A framework for faster porting of scientific applications between heterogeneous clouds. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, pp. 27–43. Springer, Cham (2018)

19. Alyahya, H., Mehmood, R., Katib, I.: Parallel sparse matrix vector multiplication on intel MIC: performance analysis. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, pp. 306–322. Springer, Cham (2018)
20. Arfat, Y., Mehmood, R., Albeshri, A.: Parallel shortest path graph computations of United States road network data on apache spark. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, pp. 323–336. Springer, Cham (2018)
21. Mehmood, R., Faisal, M.A., Altowaijri, S.: Future Networked Healthcare Systems: a review and case study. In: Boucadair, M., Jacquenet, C. (eds.) Handbook of Research on Redesigning the Future of Internet Architectures, pp. 531–558. IGI Global, Hershey (2015)
22. Alam, F., Mehmood, R., Katib, I., Albogami, N.N., Albeshri, A.: Data fusion and IoT for smart ubiquitous environments: a survey. http://ieeexplore.ieee.org/document/7911293/ (2017)
23. Muhammed, T., Mehmood, R., Albeshri, A., Katib, I.: UbeHealth: a personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities. https://ieeexplore.ieee.org/document/8382164/ (2018)
24. Suma, S., Mehmood, R., Albugami, N., Katib, I., Albeshri, A.: Enabling next generation logistics and planning for smarter societies. Procedia Comput. Sci. **109**, 1122–1127 (2017)
25. Mehmood, R., Alam, F., Albogami, N.N., Katib, I., Albeshri, A., Altowaijri, S.M.: UTiLearn: a personalised ubiquitous teaching and learning system for smart societies. IEEE Access. **5**, 2615–2635 (2017)
26. Mehmood, R., Graham, G.: Big data logistics: a health-care transport capacity sharing model. Procedia Comput. Sci. **64**, 1107–1114 (2015)
27. Why do we need to license? https://www.europeandataportal.eu/elearning/en/module4/#/id/co-01. Last accessed 30 July 2018
28. Transport Data Service—Transport for London. https://tfl.gov.uk/corporate/terms-and-conditions/transport-data-service. Last accessed 30 July 2018
29. Open data and smart cities—openmind. https://www.bbvaopenmind.com/en/open-data-and-smart-cities. Last accessed 30 July 2018
30. Carbon, S., Champieux, R., McMurry, J., Winfree, L., Wyatt, L.R., Haendel, M.: A measure of open data: a metric and analysis of reusable data practices in biomedical data resources. bioRxiv. 282830 (2018)
31. Request for community partnership in data resource licensing planning. https://figshare.com/articles/Request_for_Community_partnership_in_data_resource_licensing_planning/4972709. Last accessed 30 July 2018
32. National Institutes of Health (NIH): NIH awards to test ways to store, access, share, and compute on biomedical data in the cloud., https://www.nih.gov/news-events/news-releases/nih-awards-test-ways-store-access-share-compute-biomedical-data-cloud, last accessed 30 July 2018
33. Toga, A.W., Dinov, I.D.: Sharing big biomedical data. J. Big Data. **2**, 7 (2015)
34. (Re)usable Data Project. http://reusabledata.org/. Last accessed 30 July 2018
35. Singh, P.V., Phelps, C.: Determinants of Open Source Software License Choice: a social influence perspective part of the management information systems commons (2010)
36. Gilbert, N.: Legal tussle delays launch of huge toxicity database. http://www.nature.com/doifinder/10.1038/nature.2016.19365 (2016)
37. Oxenham, S.: Legal confusion threatens to slow data science. http://www.nature.com/doifinder/10.1038/536016a (2016)
38. Showing you this map of aggregated bullfrog occurrences would be illegal—Peter Desmet. http://peterdesmet.com/posts/illegal-bullfrogs.html. Last accessed 30 July 2018
39. Kolassa, C., Rumpe, B.: The influence of the generator's license on generated artifacts
40. About the Open Source Initiative | Open Source Initiative. https://opensource.org/about. Last accessed 30 July 2018
41. Licenses by Name | Open Source Initiative. https://opensource.org/licenses/alphabetical. Last accessed 30 July 2018

42. Various licenses and comments about them. https://www.gnu.org/licenses/license-list.en.html. Last accessed 30 July 2018
43. Open Source Hardware (OSHW) Definition. http://www.oshwa.org/definition. Last accessed 30 July 2018
44. Komninos, N., Pallot, M., Schaffers, H.: Special issue on smart cities and the future internet in Europe. J. Knowl. Econ. **4**, 119–134 (2013)
45. Caragliu, A., Del, C.B., Nijkamp, P.: Smart cities in Europe (2009)
46. Tsarchopoulos, P., Komninos, N., Kakderi, C.: Accelerating the uptake of smart city applications through cloud computing. In: World Acad. Sci. Eng. Technol. Int. J. Soc. Behav. Educ. Econ. Bus. Ind Eng. pp. 129–138. Paris, France (2017)
47. Mathur, A., Choudhary, H., Vashist, P., Thies, W., Thilagam, S.: An empirical study of license violations in open source projects. In: Proceedings of the 2012 IEEE 35th Software Engineering Workshop, SEW 2012. IEEE, pp. 168–176 (2012)
48. The Licence Proliferation Project | Open Source Initiative. https://opensource.org/proliferation. Last accessed 30 July 2018
49. German, D.M., Hassan, A.E.: License integration patterns: addressing license mismatches in component-based development. In: 2009 IEEE 31st International Conference on Software Engineering. IEEE, pp. 188–198 (2009)
50. Lindman, J., Rossi, M., Puustell, A.: Matching Open Source Software Licenses with Corresponding Business Models. IEEE Softw. **28**, 31–35 (2011)
51. Licence Assistant - European Data Portal. https://www.europeandataportal.eu/en/licence-assistant. Last accessed 30 July 2018
52. St. Laurent, A.M.S.: understanding open source and free software licensing. Ariadne. **193** (2004)
53. Kechagia, M., Spinellis, D., Androutsellis-Theotokis, S.: Open Source Licensing Across Package Dependencies. In: 2010 14th Panhellenic Conference on Informatics. IEEE (2010)
54. Fitzgerald, B.: The transformation of open source software. MIS Q. **30**, 587 (2006)
55. Kapitsaki, G.M., Tselikas, N.D., Foukarakis, I.E.: An insight into license tools for open source software systems. J. Syst. Softw. **102**, 72–87 (2015)
56. Reincke, K., Dobson, J., Sharpe, G., Schichl, P., Kern, M.: Open Source License Compendium. Version 0 (2012)
57. Heikkilä, M., Pikkarainen, J., Väisänen, M.: From proprietary to open source—Ensuring successful transition on code release
58. Monty Widenius, M., Nyman, L.: The business of open source software: a primer. Technol. Innov. Manag. Rev. 4–11 (2014)
59. Amiri-Kordestani, M., Bourdoucen, H.: A survey on embedded open source system software for the internet of things. In: FOSSC'2017, pp. 27–32 (2017)
60. Licensing milestone for data access in GBIF.org. https://www.gbif.org/news/82812/licensing-milestone-for-data-access-in-gbiforg. Last accessed 30 July 2018
61. Analyzing the licenses of all 11,000+ GBIF registered datasets—Peter Desmet. http://peterdesmet.com/posts/analyzing-gbif-data-licenses.html. Last accessed 30 July 2018
62. Chen, Y.: Business models based on open innovation strategy. In: 2011 International Conference on Management Service Science, pp. 1–4 (2011)
63. Global Open Data Index. https://index.okfn.org. Last accessed 30 July 2018
64. Open Government Licence. http://www.nationalarchives.gov.uk/doc/open-government-licence/version/2. Last accessed 30 July 2018
65. Open License / Open License | Etalab's blog. https://www.etalab.gouv.fr/licence-ouverte-open-licence. Last accessed 30 July 2018
66. About | Open Data Commons. https://opendatacommons.org/about. Last accessed 30 July 2018
67. ODC Public Domain Dedication and License Summary | Open Data Commons. https://opendatacommons.org/licenses/pddl/summary. Last accessed 30 July 2018
68. ODC Open Database License (ODbL) Summary | Open Data Commons. https://opendatacommons.org/licenses/odbl/summary. Last accessed 30 July 2018

69. Top Open Source Licenses | Black Duck Software. https://www.blackducksoftware.com/top-open-source-licenses. Last accessed 30 July 2018
70. SPDX License List | Software Package Data Exchange (SPDX). https://spdx.org/licenses. Last accessed 30 July 2018
71. Apache License, Version 2.0. https://www.apache.org/licenses/LICENSE-2.0. Last accessed 30 July 2018
72. Goldman, R., Gabriel, R.P., Kaufmann, M.: Innovation Happens Elsewhere: Open Source as Business Strategy. Morgan Kaufmann, Burlington (2005)
73. Eclipse Public License 1.0 (EPL) Frequently Asked Questions | The Eclipse Foundation. https://www.eclipse.org/legal/eplfaq.php. Last accessed 30 July 2018
74. EPL-2.0 FAQ | The Eclipse Foundation. https://www.eclipse.org/legal/epl-2.0/faq.php. Last accessed 30 July 2018
75. Community Data License Agreement—Permissive, Version 1.0—CDLA. https://cdla.io/permissive-1-0. Last accessed 30 July 2018
76. Community Data License Agreement—Sharing, Version 1.0—CDLA. https://cdla.io/sharing-1-0. Last accessed 30 July 2018
77. Fogel, K.: Producing open source software: how to run a successful free software project. O'Reilly, Newton (2005)

Chapter 23
Big Data and HPC Convergence for Smart Infrastructures: A Review and Proposed Architecture

Sardar Usman, Rashid Mehmood, and Iyad Katib

23.1 Introduction

The data growth over the last couple of decades increases on a massive scale. As the volume of the data increases so are the challenges associated with big data. Over the years, high performance computing (HPC) has contributed a lot in scientific discoveries, improved engineering designs, enhanced manufacturing, fraud detection, health care, and national security, thus played crucial role towards quality of human life. The world has seen exponential data growth due to social media, mobility, E-commerce, and other factors. Major chunk of data has been generated in the last few years alone and is even growing at more rapid rate [1]. To deal with ever-growing volume of data, researchers have been involved in developing algorithms to accelerate the extraction of key information from massive data. Big data is a buzzword, which has caught a great deal of attention in the recent years. Big data refers to "the emerging technologies that are designed to extract value from data having four Vs characteristics, volume, variety, velocity, and veracity" [2]. It usually implies massive amounts of structured, semi-structured, and unstructured data collected from different resources and is not possible to store and process this data by traditional databases and software techniques.

Historically only the largest companies, government research organizations, and academic computing centers have had an access to the computing power necessary to get to valuable conclusions in a reasonable amount of time. All that is rapidly

S. Usman (✉) · I. Katib
Department of Computer Science, FCIT, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: susman@stu.kau.edu.sa; iakatib@kau.edu.sa

R. Mehmood
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: RMehmood@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_23

changing with vast improvement in the price, performance, availability, and density of compute power beyond the human imagination.

The categorization of data vs. computing is affected by solution urgency, i.e., real-time solution, and also depends on what we are trying to achieve. As the volume of data is growing bigger, it brings more challenges to process that data in real time. As projected, in 2018 over 4.3 exabyte of data will be created on daily basis [3]. Over the years HPC community has not been deprived of huge volume of data, i.e., climate modeling, design and manufacturing, and financial services that resulted in high fidelity models and interdisciplinary analysis to explore data for deeper insights. The use of High Performance Data Analytics (HPDA) is increasing at brisk speed in many industries, resulting in expansion of HPC market in these new territories.

Powerful analytics is a key to extract a value from data by confronting budget and marketing challenges and plays huge roles in making plans, predicting business trends, and understanding customer demands. Choosing a right solution depends on the size of data, urgency of results, prediction about the needs of more processing power as the size of data increases, fault tolerance for applications in case of hardware failure, data rate, scalability, etc. A real-time application with high response time especially when dealing with huge volume of data is still a challenging task and is one of the driving forces towards the convergence of big data and HPC.

Both HPC and big data are different systems not only at the technical level but also have two different ecosystems. Both have different programming models, resource managers, file systems, and hardware. HPC is mainly developed for computational intensive applications but recently data intensive applications are also among the major workloads in HPC environments. Big data is relatively a new term compared to HPC that has been in research for decades (see, for example, [4–13]). Due to recent advancements of data intensive applications, number of software frameworks has been developed for distributed systems, cluster resource management, parallel programming models, and machine learning frameworks. High performance computing has very well established standard programming model, e.g., Open MP/MPI. Big data analytics have been grown up in different perspective and have different population of developers that uses java and other high level languages with primary focus on simplicity of use, so that problem domain can be solved without a detailed knowledge of HPC. These differences in the infrastructure, resource manager, file system, and hardware make the system integration a challenging task.

As the data is getting bigger and bigger in volume so is the need of high computing. HPC community has been dealing with massive amount of data and big data analytics for years. The solutions evolved over the years to deal with large volume of data should be useful for big data analytics [14]. The main aim of this chapter is to identify motivation and driving forces towards the integration of HPC and big data and also highlight the current trends, challenges, benefits, and future aspects of unified integrated system. We also present architecture for the convergence of HPC and big data using design patterns. This chapter is an extension

of our earlier work [15]. We have added a whole new section on programming models and frameworks of big data and HPC. Another new section on the big data and HPC challenges in the exascale-computing era has been added. Further elaborations are provided on HPC and big data convergence research efforts. The proposed HPC-big data convergence architecture has been enhanced.

The rest of the chapter is organized as follows. The next section gives a brief overview of some of the most widely used HPC and big data models and frameworks. Section 23.3 examines the difference between HPC and Hadoop framework with respect to hardware, resource management, fault tolerance, and programming model. Section 23.4 highlights some of the big data and HPC challenges in the exascale era. Literature survey is presented in Sect. 23.5 and convergence challenges are discussed in Sect. 23.6 followed by the future directions in Sect. 23.7. The proposed architecture using design pattern for the convergence of HPC and big data is presented in Sect. 23.8 and chapter is concluded in the Sect. 23.9.

23.2 HPC and Big Data Programming Models and Frameworks

This section gives a brief overview of some of the most widely used HPC and big data frameworks and programming interfaces.

23.2.1 Hadoop

Hadoop is an open source Apache project out of yahoo in 2006, which provides distributed fault-tolerant data storage, batch processing, and linear scalability on commodity hardware. Hadoop is adopted successfully by many organizations like Amazon, AOL, EBay, Facebook, and Twitter. Hadoop can be broadly categorized into two units, i.e., Hadoop distributed file system (HDFS) and map-reduce processing [16]. The core concept behind Hadoop is to divide a job into multiple tasks to be executed on the multiple data processing nodes. HDFS is used to store data with high availability to multiple nodes (Fig. 23.1).

Hadoop is shared nothing architecture, where each node acts independently throughout the system and a piece of work is divided among several parallel map/reduce tasks. Each task operates independently on cheap commodity servers. This enables businesses to generate value for data that was previously considered too expensive to store and process in traditional data warehouse or OLTP environment. In the old paradigm, companies were using traditional enterprise data warehouse system and would buy biggest data warehouse that they could afford and store data on a single machine. However with the increased amount of data being generated

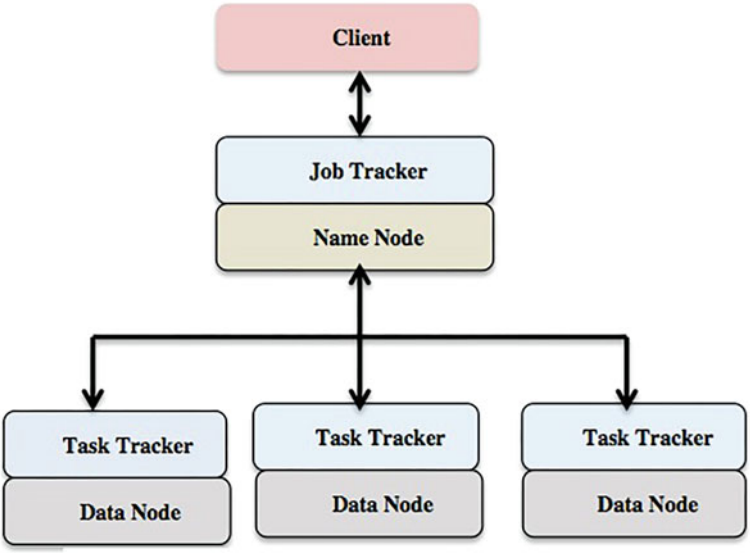


Fig. 23.1 Hadoop architecture overview

every day, this is no longer affordable or practical. The traditional enterprise data warehouse may not be able to address big data problem space [1].

23.2.2 Spark

Both Hadoop and Spark are big data frameworks and do perform the same tasks, are not mutually exclusive and able to work together. Spark is mostly used on the top of Hadoop and advance analytics of spark are used on data stored in Hadoop's distributed file system (HDFS). Spark has the ability to run as Hadoop's module through YARN and as a standalone solution [17] and can be seen as an alternative to map-reduce rather than a replacement to Hadoop framework (Fig. 23.2).

Spark is much faster compared to Hadoop because it handles in-memory operations by copying data from distributed file systems into faster logical RAM. Map-reduce writes all data back to distributed storage system after each iteration to ensure full recovery, whereas Spark arranges data in resilient distributed data sets that are capable of full recovery in case of failure. Spark capability of handling advance data analytics in real-time stream processing and machine learning is a much more advance that gives Spark edge over Hadoop. The choice of selecting either of the data processing tool depends on the needs of an organization, e.g., dealing with big structured data can be done efficiently with map-reduce and there is no need to install a separate layer of Spark over Hadoop [18]. Spark on demand allows users to use Apache Spark for in situ data analysis of big data on HPC

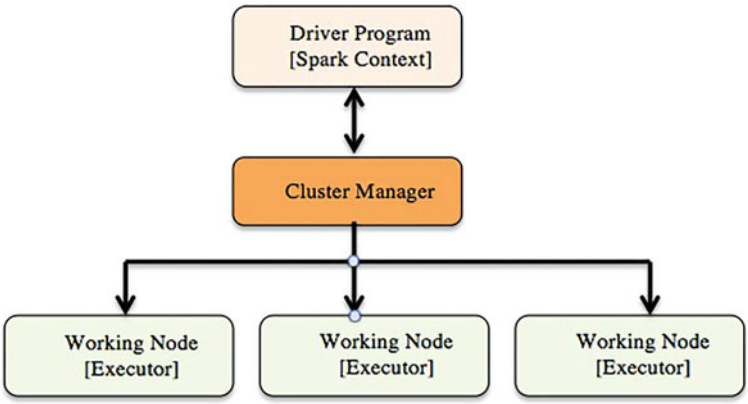


Fig. 23.2 Spark architecture overview

resources [19]. With this setup, there is no longer need to move petabytes of data for advance data analytics.

23.2.3 MPI (Message-Passing Interface)

MPI is widely a de facto standard for writing message-passing programs and provides standard programming model where messages communicate with each other by using different routines to send and receive messages in distributed/shared memory systems. The Message-Passing Interface (MPI) is a standardization of a message-passing library interface specification. MPI defines the syntax and semantics of library routines for standard communication patterns. MPI provides user a complete control to optimize data distribution, synchronization and enhance data locality [20]. The programmer is responsible for handling all issues related to parallelism, i.e., parallel algorithms, data structures, load balancing, communication commands, and synchronization (Fig. 23.3).

23.2.4 OpenMP

OpenMP (Open Multiprocessing) targets shared memory systems used for multi-threaded parallel processing available in gcc (gnu compiler), Intel compiler, and others. OpenMP directives can be inserted into C/C++ or Fortran programs and have runtime library for organizing parallel parameters. The environment variables and library routines control the runtime system (Fig. 23.4).

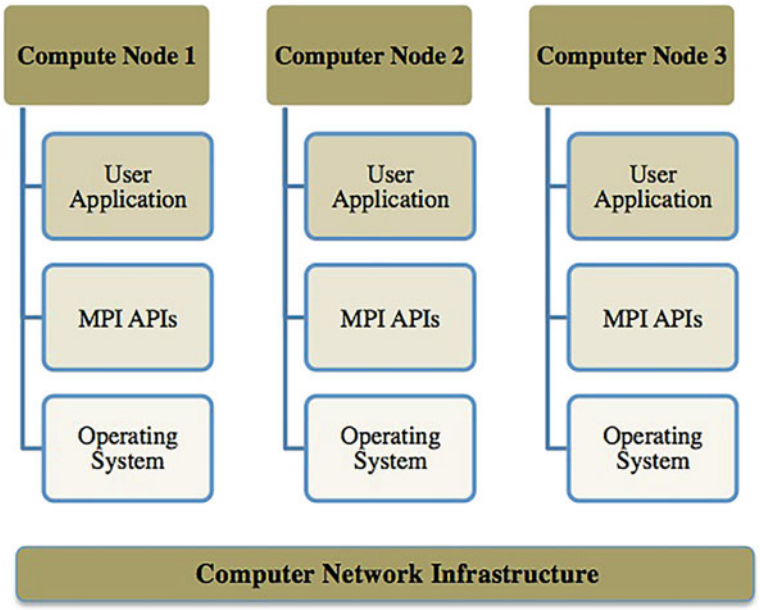


Fig. 23.3 MPI application architecture

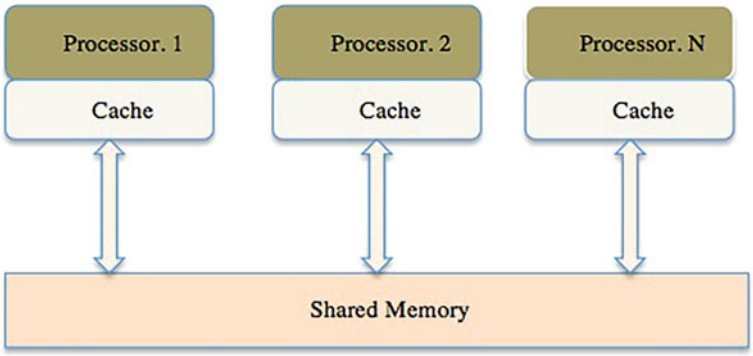


Fig. 23.4 Shared memory

23.2.5 PGAS

PGAS (Partitioned Global Address Space) is a parallel programming model where global memory is logically shared among processes/threads, which improves programmer productivity without compromising high performance [21]. The global address space is partitioned among the cluster nodes and is basis of Co-array Fortran, Split-C, X10, Unified Parallel C, etc. [22] (Fig. 23.5).

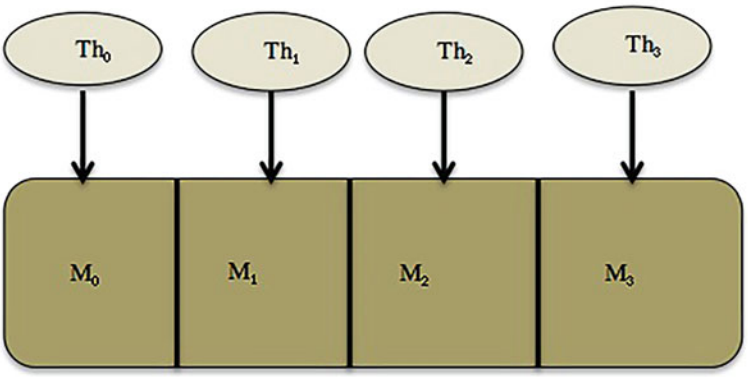


Fig. 23.5 Partitioned global address space

23.3 HPC and Big Data Frameworks and Their Differences

Different solutions emerged over the years to deal with big data issues and are successfully implemented. But never the less, all these solutions do not satisfy the ever-growing needs of big data. The issues related to big data are immense and cover variety of challenges that needs a careful consideration, for example data representation, data reduction/compression, data confidentiality, energy management, high dimensionality, scalability, real and distributed computation, non-structured processing, analytical mechanism, and computational complexity. The exponential outburst of data and rapidly increasing demands for real-time analytical solutions urges the need for the convergence of high-end commercial analytics and HPC. Business intelligence/analytical solutions today, lack the support for predictive analytics, lack of data granularity, lack of software flexibility to manipulate data, lack of intuitive user interface, relevant information is not aggregated in a required manner and have slow system performance [23].

HPC community has been dealing with complex data and compute intensive applications, and solutions have been evolved over the years. As the volume of data is increasing at brisk speed so are the associated challenges, i.e., data analysis, minimizing data movement, data storage, data locality, and efficient searching. As we are heading towards exascale era, the increase in system concurrency introduced a massive challenge for system software to manage applications to perform at extreme level of parallelism. Large-scale applications use most widely deployed message-passing programming model MPI along with traditional sequential languages, but with the introduction of architectural changes (many core chip) and high demand in parallelism make this programming model less productive for exascale systems. Billion-fold parallelism is required to exploit the performance of extreme scale machines and locality is critical in terms of energy consumption. As the complexity and scale of software requirements is on a rise, simple execution model is a critical requirement, which ultimately reduce the application programming

complexity required to achieve the goals of achieving extreme scale parallelism. A current trend in HPC market includes the use of advanced interconnects and RDMA protocols (Infinity Band, 10–40 Gigabits Ethernet/iWARP, RDMA over converged Enhanced Ethernet), enhanced redesign of HPC middleware (MPI, PGAS), SSDs, NVRAM, burst buffer, etc. Scalable parallelism, synchronization, minimizing communication, task scheduling, memory wall, heterogeneous architecture, fault tolerance, software sustainability, memory latencies, simple execution environment, and dynamic memory access for data intensive application are some of the core areas that requires considerable time and efforts to address exascale challenges [24]. The difference between Hadoop and HPC framework is highlighted in the following section.

23.3.1 Hardware

Most of the modern HPC and Hadoop clusters are commodity hardware. In HPC environment, compute nodes are separated from data nodes. There are two types of data storage, temporal file system on local nodes and persistent global shared parallel file system on data nodes. The existing HPC clusters have limited amount of storage on each compute node. LUSTRE is most widely used parallel file system in HPC and almost 60% of the top 500 supercomputers use LUSTRE as their persistent storage. Data needs to be transferred from data nodes to the local file system on each compute node for processing. Data sharing is easy with distinct data and compute nodes but spatial locality of data is an issue [25, 26].

Hadoop cluster uses local disk space as a primary storage. The same node serves as a data node and compute node. The computational task is scheduled on same machine where data is resided resulting in enhanced data locality. Hadoop is write-once and read-many framework. I/O throughput of Hadoop is much higher, due to co-locating of data and compute node on the same machine [26].

23.3.2 Resource Management

Another major difference between Hadoop and HPC cluster is resource management. Hadoop's name node has job tracker daemon. Job tracker supervised all map-reduce tasks and communicates with the task trackers on the data node. Compared to Hadoop's integrated job scheduler, HPC scheduling is done with the help of specialized tools like grid engine and load leveler [27] with controlled resources (memory, time) provided to the user.

23.3.3 Fault Tolerance

HPC resource scheduler uses checkpoint mechanism for fault tolerance. In case of node failure, it reschedules job from the last stored checkpoint. It needs to restart the whole process if the checkpoint mechanism is not used. On the other hand, Hadoop uses job tracker for fault tolerance. As data and computation are co-located on same machine, job tracker can detect a node failure on run time by re-assigning a task on a node where duplicate copy of data is resided [27, 28].

23.3.4 Programming Model

Hadoop uses map-reduce programming model, which makes life easier for the programmers as they just need to define map step and reduce step, when compared to the programming efforts needed for HPC applications. In HPC environment, programmer needs to take fine-grained responsibilities of managing communication, I/O, debugging, synchronization, and checkpoint mechanism. All these tasks need considerable amount of efforts and time for effective and efficient implementation. Hadoop does provide a low level interface to write and run map-reduce applications written in any language, although Hadoop is written in Java. Table 23.1 summarizes the difference between HPC and Hadoop framework [26].

Table 23.1 HPC vs. Hadoop Ecosystem

	Big data	HPC
Programming model	Java applications, SparQL	Fortran, C, C++
High-level programming	Pig, Hive, Drill	Domain specific language
Parallel runtime	MapReduce	MPI, Open MP, OpenCL
Data management	HBase, MySQL	iRODS
Scheduling (resource management)	YARN	SLURM (Simple LINUX Utility for Resource Management)
File system	HDFS, SPARK (local storage)	LUSTRE (remote storage)
Storage	Local shared nothing architecture	Remote shared parallel storage
Hardware for storage	HDDS	SSD
Interconnect	Switch Ethernet	Switch Fibre
Infrastructure	Cloud	Supercomputer

23.4 Big Data and HPC Challenges in the Exascale-Computing Era

The following section briefly introduces some of the big data issues and also highlights the exascale challenges.

23.4.1 Data Heterogeneity

One of the biggest challenges of big data is to deal with data heterogeneity. As data is collected from different resources, which is, either structure, semi-structure or unstructured having different semantics, granularity, and accessibility. Effective data analysis will be a challenging task with improper data representation. Data needs to be pre-processed for efficient representation, improved quality data access, and effective analytical results. To perform the big data acquisition, organizational strategy must be considered for data ownership, data value, and quality assurance [29].

23.4.2 Data Redundancy

Effective data redundancy reduction and compression techniques need to be implemented as most data from heterogeneous resources like sensor networks is highly redundant.

23.4.3 Data Freshness

Data freshness is another important aspect, as values encapsulated in big data depend on data freshness. There is need for comprehensive mechanism to decide which data needs to be stored and which data needs to be discarded [30].

23.4.4 Data Confidentiality

Most of the organizations could not store and process their data due to their limited capacity and are heavily dependent on the third-party solutions, which give rise to another fundamental problem of data confidentiality [30].

23.4.5 Big Data Access and Processing

Processing the data for data mining and interpretation is a challenging task due to diverse nature of big data. Over the years, numerous data mining and machine learning algorithms have been constructed but mainly for the structured data and lack the ability to work on semi-structured or unstructured data [29]. There is a need for a comprehensive mechanism to construct these semi-structured or unstructured data sets into structured format and then expose them to well-known structured data mining or machine learning algorithms.

23.4.6 Scalability

As the rate at which the data is being generated is increasing exponentially, scalability is another major concern in big data. Analysis of big data needs to be performed in such a way that supports complex present and future data sets.

23.4.7 Engineering

Another fundamental challenge is how to effectively discover knowledge from big data analysis and efficiently use that knowledge in a real-life application. This engineering problem perhaps depends on the efficient co-operation from interdisciplinary domain experts [31].

23.4.8 Data Storage

The digital data growth has been immense and according to IDC [32] estimate by 2020, 2.5 exabyte of data will be created every day which resulted in a challenging task of how we can efficiently store and make data available for further analysis.

Table 23.2 summarizes some of the challenges faced by big data and HPC community.

23.5 Research Related to HPC and Big data Convergence

The integration of HPC and big data started at different levels of their ecosystems and these integrated solutions are still at very infant stages. The convergence of both these technologies is the hottest topic for the researchers over the last few

Table 23.2 Exascale, big data and business/analytical challenges

Exascale challenges	Big data challenges	Business/analytical solutions (challenges)
Data analysis	Data representation	Lack of predictive analytics
Minimizing data movement	Data reduction	Lack of data granularity
Data storage	Data confidentiality	Lack of software flexibility to manipulate data
Data locality	Energy management	Lack of intuitive user interface
Fault tolerance	High dimensionality	Slow system performance
Energy efficiency	Real and distributed computing	
Scalable parallelism	Unstructured processing	
Synchronization	Analytical mechanism	
Memory wall	Computational complexity	
Heterogeneity		
Software sustainability		
Simple execution environment		
Dynamic memory access for data intensive applications		

years. Reed et al. [33] provided a comprehensive view of some of the technical challenges, global ecosystems, advancement in computing, and relationship between data analysis and computational modeling. They highlighted some of the benefits of advanced computing from biology and medicines to High Energy Physics (HEP), climate science, cosmology, astrophysics, experimental and computational material sciences, etc. They also highlighted some of the technical challenges in advanced computing including hardware and architecture challenges, resilience and energy efficiency at scale, locality and scalability, parallel programming support, algorithmic and mathematics challenges, economic and political challenges, etc. Several international workshops have been organized over the years to address the issues and challenges faced by the inevitable convergence of the two paradigms; see for example the Big Data and Extreme-scale Computing (BDEC) project report [34]. BDEC community focused on the problems of two different divisions of ecosystem, i.e., the split between HPC & HPDA and stateless network & state-full service provided by the end systems. They focused on the most critical big data challenges, i.e., logistics of wide area, data productions, data transformation, data sharing and analysis. The software infrastructure needs to be reframed to accommodate the diversity of the workflow patterns. They discussed the opportunities and challenges to converge HPC and HPDA, and also highlighted the problems associated with the converged infrastructure for distributed edge environments. The authors also discussed about the opportunities for the converged software ecosystem, specifically focused on logical centralized facility capable of running both high-end simulations and large-scale data analytics [34].

In [25] Krishnan et al. proposed a myHadoop framework using standard batch scheduling system for configuring Hadoop on-demand on traditional HPC resources. The overhead in this setup includes site-specific configuration, keeping input data into HDFS and then staging results back to persistent storage. HDFS is heavily criticized for its I/O bottleneck. Availability of limited storage is big challenge to integrate Hadoop with HPC clusters. Islam et al. [35] proposed a hybrid design (Triple-H) to reduce I/O bottleneck in HDFS and efficient resource utilization for different analytics system performance and cluster efficiency with overall low system cost.

Data intensive applications have been intensively used in HPC infrastructure with multicore systems using map-reduce programming model [36]. With increase in parallelism, the overall throughput increases resulted in high energy efficiency as the task is completed in shorter span of time. When Hadoop runs on HPC cluster with multiple cores and each node is capable of running many map/reduce tasks using these cores. This ultimately decreases the data movement cost and increases throughput but due to high disk and network accesses of map-reduce tasks, the energy consumption and throughput cannot be predicted. High degree of parallelism may or may not affect energy efficiency and high performance. MPI (Message-Passing Interface) is most commonly used parallel programming paradigm to exploit parallelism and rely on the high-speed/low-latency network for message passing and make use of parallel file systems, e.g., Lustre [37], GPFS [38], and PVS [39]. Data intensive computing makes use of distributed file systems, which includes Google file system GFS [38], HDFS Hadoop distributed file system [40], etc. The HPC applications use data intensive distributed file system through an interface for example libHDFS [41]. Over the years, different solution started to emerge to cope with massive amount of data more specifically in data intensive applications, e.g., SRM [42], iRODS [43], MapReduce-MPI [44], and Pilot-MapReduce [45].

Tiwari et al. [46] studied the Hadoop's energy efficiency on HPC cluster. Their study shows that energy efficiency of map-reduce job on HPC cluster changes with increase in parallelism and network bandwidth. They determine the degree of parallelism on a node for improving the energy efficiency and also benefits of increasing the network bandwidth on energy efficiency by selecting configuration parameters on different types of workloads, i.e., CPU intensive and moderate I/O intensive, CPU and I/O intensive workloads, also energy and performance characteristics of a disk and network I/O intensive jobs. When the number of map slots reached beyond 40, number of killed map tasks almost doubled. Thus increasing the parallelism to certain extent has positive impact on energy efficiency.

Scientific data sets are stored in back end storage servers in HPC environment and these data sets can be analyzed by YARN map-reduce program on compute nodes. As both compute and storage servers are separated in HPC environment, the cost of moving these large data sets is very high. The high-end computation machine and analysis clusters are connected with high-speed parallel file system. To overcome the shortcomings of offline data analysis, "in situ" data analysis can be performed on output data before it is written to parallel file system. The use high-end computation node for data analysis results in slowing down simulation job by

Table 23.3 Current convergence efforts

Approaches	Solutions
iRODS, MapReduce-MPI, Pilot-MapReduce, SRM, etc.	Solutions to deal with massive amount of data in data intensive applications
Spark on demand	Apache Spark for in situ data analysis of big data on HPC resources
Mellanox UDA, RDMA-Hadoop, DataMPI, Hadoop with IPoIB (IP over InfiniBand), HMOR, etc.	HPC-oriented map-reduce solutions
myHadoop	Hadoop on-demand on traditional HPC resources
LibHDFS	HPC application's interface to HDFS
MPI, ad-hoc Hadoop, CloudBlast, Spark, HTCondor	Parallelization of many task applications with different workflow systems
DataMPI	A framework which exploits the overlapping of map, shuffle and merge phases of map-reduce framework
Virtualized Analytics Shipping (VAS), Spark on demand	Map-reduce-like framework for in situ data analysis

the interference of the analysis task and inefficient use of computation resources for data analysis tasks. Spark on demand allows users to use Apache Spark for in situ data analysis of big data on HPC resources [19]. With this setup, there is no longer to move petabytes of data for advance data analytics.

According to Dhabaleshwar K. Panda [47], the use of InfiniBand for large clusters is most cost effective than standard Ethernet. The performance of HPC oriented map-reduce solutions (Mellanox UDA, RDMA-Hadoop, DataMPI, etc.) depends on the degree of change in Hadoop framework as more deep modification means an optimal adaption to HPC systems. Hadoop with IPoIB (IP over InfiniBand) and Mellanox UDA requires minimal or no changes in Hadoop implementation and only requires minor changes in Hadoop configuration. RDMA-Hadoop and HMOR are the HPC oriented solutions to take advantage of high-speed interconnects by modifying some of the subsystems of Hadoop. DataMPI is a framework that developed from the scratch, which exploits the overlapping of map, shuffle, and merge phases of map-reduce framework and increases data locality during the reduce phase. DataMPI provides the best performance and an average energy efficiency [48]. Table 23.3 gives the brief overview of some of the efforts, which one can relate towards the convergence of HPC and big data.

The use of InfiniBand improved the network bandwidth, as InfiniBand being widely used in HPC environment. Communication support in Hadoop relies on TCP/IP protocol through Java sockets [48]. So it is difficult to use high performance interconnects in an optimal way so different HPC oriented map-reduce solutions came that addresses the problem of leveraging high performance interconnects RDMA-Hadoop, DataMPI, etc. Yandang et al. [49] compared the performance of 10 GigaBit Ethernet and InfiniBand on Hadoop. With small intermediate data sizes, the

use of high-speed interconnect increased the performance by efficiently accelerating jobs but doesn't show the same performance with large intermediate data size. The use of InfiniBand on Hadoop provides better scalability and removes the disk bottleneck issues. As the Hadoop cluster is getting bigger, organizations feel the need of specialized gear like solid-state drives (SSDs) and the use of InfiniBand instead of standard Ethernet. The use of InfiniBand with RDMA (remote direct memory access) allows 40 Gigabits/s raw capacity out of Quad Data Rate (QDR) InfiniBand port which is four times as much bandwidth as 10 GigaBit Ethernet port can deliver [47].

The use of InfiniBand allows maximum scalability and performance while overcoming the bottlenecks in the I/O. Islam et al. [50] proposes an alternative parallel replication scheme compared to pipelined fashioned replication scheme by analyzing the challenges and compared its performance with existing pipelined replication in HDFS over Ethernet, IPoIB, 10 GigE and RDMA and showed performance enhancement with parallel model for large data sizes and high performance interconnects.

23.6 Challenges of HPC-Big Data Convergence

The world of workload is diverse enough and performance sensitivity is high enough that we cannot have globally optimal and local high sub-optimal solution to all the issues related to convergence of HPC and big data. HPC and big data architectures are different and have different ecosystem. The cross-fertilization of HPC and big data is the hottest topic for the researchers over the last few years. Most of the research related to the convergence of HPC and big data started at distinct levels of ecosystem but do not address the problem of moving data especially in HPC environment. The integration of data intensive applications in HPC environment will bring many challenges. In exasacle environment, cost of moving big data will be more than cost of floating point operations. There is a need for high energy efficient and cost effective interconnects for high bandwidth data exchange among thousands of processors. We also need a data locality aware mechanism especially when dealing with big data in HPC shared memory architecture. The cost of moving big data for processing also brings another challenge of high power consumption. With massively parallel architecture with hundreds of thousands of processing nodes, the cost of moving data will be very high. According to Keckler et al. [51], energy efficiency of 20 pJ (Pico Joules) per floating point operation is required for exascale system, whereas current state of art multicore CPUs have 1700 pJ and GPUs have 225 pJ per floating point operation.

Minimizing the data movements means the innovation in memory technologies with enhanced capacity and bandwidth. To deal with 3Vs (volume, velocity, veracity) of big data, efficient data management techniques need to be investigated including data mining and data co-ordination [35] as most of the HPC platforms are compute centric, as opposed to the demands of big data (continuous processing,

efficient movement of data between storage devices, network connections, etc.). To deal with massive parallel architecture and heterogeneous nature of big data, innovation is needed at the programming model to deal with the next generation of parallel systems, thus reducing the burden of parallelism and data locality for application developer as MPI leaves it to the programmer to handle issues related to parallelism. Hadoop being widely used as a big data framework achieves fault tolerance by the replication of data on multiple nodes and job tracker assigns job to other node in case of node failure. Fault tolerance in HPC is by means of checkpoint mechanism, which is heavily criticized and not suitable for exascale environment. In exascale systems, hardware failure will be a rule not an exception. The MTBF (mean time between failures) window in current peta-scale system is in days and for exascale systems it will be in minutes or may be few seconds. So there is need for a comprehensive resilience at the different levels of exascale ecosystem. Exascale systems will be constrained by power consumption, memory per core, data movement cost, and fault tolerance. The integration between HPC and big data must address the issues of scalability, fault resilience, energy efficiency, scientific productivity, programmability, and performance [33].

Resilience, power consumption, and performance are inter-related to each other. High degree of resilience or fault tolerance is achieved but on the expense of high power consumption. As we are heading towards exascale era, convergence of both HPC and big data will make energy efficiency a core issue to handle. Servers and data-centers are facing the same problem of power consumption including companies like Google, Amazon, and Facebook. According to an estimate, the actual cost of exascale system will be less than cost of power consumption for maintaining and running exascale system for 1 year [52].

The energy efficiency techniques in big data can be broadly categorized as software/hardware based energy efficient techniques, energy efficient algorithms and architectures. A set of commodity hardware is used in both HPC and big data platforms for processing of data. The integrated hardware solution for data intensive applications and computational intensive applications wouldn't work for exascale systems as hardware solution helps to achieve fault tolerance but on the expense of high energy consumption. The current peta-scale high performance computing with checkpoint mechanism to achieve fault tolerance and energy efficiency does not suit well for the integrated solution of HPC (exascale) and big data. Soft, hard, and silent errors in exascale environment will be rule not an exception. Thus collaborative efforts are needed at system level or application level resilience to deal with fault tolerance and energy efficiency for the integrated solution.

As we have seen that both HPC and Hadoop (big data) architectures are different and have different ecosystem. Both have different programming model, resource manager, file system, and hardware. These differences in the infrastructure, resource manager, file system, and hardware make the system integration a challenging task. As the data is getting bigger and bigger in volume so is the need of high computing. One of the biggest challenges that both big data and HPC community facing is energy efficiency. Exascale parallel computing system will have thousands of nodes with hundreds of cores each and is projected to have billions of threads of execution.

The frame of Main Time Between Failures (MTBF) in supercomputers is in days and weeks. But for exascale computing with million times more components, the perception of MTBF is in hours or minutes or may be in seconds. Each layer of exascale ecosystem must be able to cope with the errors [53].

Real-time data analysis is also a driving force behind the urgency of the need for the necessary convergence of the analytics, big data, and HPC when dealing with computation, storage and analysis of massive, complex data sets in high scalable environment. Scalability issues should be addressed by the HPC community, by capitalizing the advancements in network technologies (low-latency network), also efficient and large memory should also address the scalability issues of the data analytics [54].

23.7 Driving Forces and Future Aspects

High performance data analytics (HPDA) includes tasks involving massive amount of structured, semi-structured, and unstructured data volumes and highly complex algorithms that ultimately demands the needs of HPC resources.

Companies now have the computing power they need to actually analyze and act upon their data. This translates into numerous benefits for the company, environment, and society over all. In the energy sector, companies are now able to more accurately drill for oil. Automobiles and airlines are much safer due to rapid modeling of operational data design optimization and aerodynamics analysis, allowing them to deliver more cost effective products that operate safer and are more fuel-efficient. In the financial sector, banks and card issuers can do fraud detection in real time. Stock investors can quickly track trends in the market to better serve their investing customers. Retailers and advertisers can now review historic purchasing data to better deliver the right products and advertisement to their customers and whether researchers can study thousands of years of weather data in hours or days instead of weeks or months, improving the quality of predictions and safety of people worldwide. HPC industry has been dealing with data intensive simulations and high performance analytics solutions also evolved over the years urge the commercial organizations to adopt HPC technology for competitive advantage to deal with time critical and highly variable complex problems. The chasm between data and compute power is becoming smaller all the time. The global HPDA market is growing rapidly and according to forecast HPDA global market size was US 25.2 billion and with the growth of nearly 18%, it is projected to be around US 82 billion in 2022 [55].

Fault tolerance, high power consumption, data centric processing, limitations of I/O, and memory performance are few of the driving forces that are reshaping the HPC platforms to achieve exascale computing [56]. Data intensive simulations, complex and time critical data analytics require high performance data analytics solutions for example intelligence community, data driven science/engineering, machine learning, deep learning, knowledge discovery, etc. These competitive

forces have pushed relatively new commercial companies (Small and Medium scale Enterprises—SMEs) into HPC competency space. Fraud/anomaly detection, affinity marketing, business intelligence, and precision medicine are some of the perusable new commercial HPC market segments that require high performance data analytics. The use of HPDA will increase with time in future demanding convergence of HPC and big data. HPDA is becoming an integral part of future business investments plans of enterprises, to enhance customer experience, anomaly detection marketing, business intelligence, security breaches, etc. and discovery of new revenue opportunities.

23.7.1 The Internet of Things (IoT) and Smart Cities

IoT links physical devices (computers, sensors, electronics,) equipped with sensors to the Internet and network connectivity enabling them to communicate. The common IoT platform brings heterogeneous information together and facilitates communication by providing common language. According to Gartner [57], IoT units installed base will reach 20.8 billion by 2020 resulted in massive amount of data which will further highlight the security, customer privacy, storage management, and data centric networks challenges. Smart city demands better and more innovative services to run whole city smoothly and improve people's life through the innovative use of data; see, e.g., smart infrastructure [15, 58], healthcare [59–62], transport [63–70], and other applications [5, 71].

Smart cities and IoT are some of the emerging HPDA application areas. HPC has been involved in managing power grids and transport for the upstream design of vehicles and urban traffic management in smart cities for quite some time and its use over time will increase in the markets of cognitive computing/AI, driverless vehicles, and healthcare organizations. Baz [72] investigated the connection between IoT and HPC by highlighting some of the challenges in smart world applications (smart building management, smart logistics, and smart manufacturing) and possible opportunities with HPC enable solutions. China's HPC-IoT plan 2030 is based on the use of HPC in IoT network wellness management and security [73].

23.7.2 Cognitive Technology

Cognitive systems are capable of understanding complex language constructs, correlate the association and help to rationalize information and discover insights. The key in cognitive systems is learning, adaptability and how the system is evolving, helps in decision-making process, discovery of new ventures, improved production and operation systems, optimizing resources, proactive identification of faults ahead of failure, etc. The motive of cognitive computing is to handle complex problems without no or little human intervention. According to IBM estimate, 80% of data is

unstructured and is of no use for the machines and not fully exploited. The cognitive computing can be seen as a potential candidate for the exploration of unstructured data to get more useful information insights and efficient decision-making. The rapid growth of data from multidisciplinary domains requires powerful analytics but lacks human expertise to tackle the diverse and complicated problems. The cognitive computing allows people with less experience to interact with machine thanks to the advancement in natural language processing and artificial intelligence technologies, e.g., Google DeepMind and Qualcomm's Zeroth Platform. The advancement in cognitive technology with the integration of AI and machine learning for big data tools and platforms will increase the quality of information, dealing with the complex data analytics with lesser human intervention but requires rapid data access (low latency), faster time to insights, hardware acceleration for complex analytics [3]. Extracting information from vast amount of data requires innovation in compute and storage technologies, which should provide cost effective storage, improved performance in a desired time frame. The infrastructure required cognitive storage with learning ability for computers to store only relevant and important data. The computing requires efficient processing which demands high memory bandwidth and extreme scale parallelism for efficient resource utilization within energy efficiency constraints. Open power foundation [3] is an initiative towards partnering technology solutions with diverse companies coming together to provide technology solutions to a variety of problems. With data centric computing, time to solution will be dramatically reduced. Cognitive computing is though still at its infancy stages but in future will be a key technology for the success of modern businesses, to get insights of the vast amount of unstructured data by leveraging computing technology to work better with the way humans want to work and smoothing the natural relationship between human and the computer.

23.8 The Proposed HPC-Big Data Convergence Architecture Based on Design Patterns

The need for HPDA demands innovative ways, to accelerate data and predictive analysis to target above-mentioned complex challenges by revolutionary and evolutionary changes in programming models, computer architecture, and runtime systems to accommodate potential interoperability and scaling convergence of HPC and big data ecosystems [3]. There is growing need for the efficient exploration of novel techniques to allow HPC and big data applications to exploit billion-fold parallelism (exascale systems), improved data locality, unified storage systems, synchronization, and ultimately the single system architecture to overcome the cost and complexity of moving data which also improves the total cost of ownership and brings in flexibility to manage workflows and maximize system utilization. Design patterns and skeletons are the potential candidates to address above-mentioned challenges to design scalable, robust software development and applicable proved solutions in both HPC and big data community.

The parallel programming problem has been an active area of research for decades focusing primarily on programming models and their supporting environments. As we move towards exascale (millions of components, billions of cores) programming parallel processors and handling billion-way parallelism is one of the major challenges that research community is facing. Software architecture and design plays a vital role in designing robust and scalable software. Common set of design elements (derived from domain expert's solutions) are captured in a design pattern of that particular domain to assist the software designer to engineer robust and scalable parallel software. These patterns define the building blocks of all software engineering and are fundamental to architect parallel software. The design problem at different level of software development is addressed by developing layered hierarchy of patterns by arranging patterns at different levels. These design patterns have been developed to assist software engineers to architect and implement parallel software efficiently. Our Pattern Language OPL is one of prominent source of cataloguing and categorizing the parallel patterns [74]. A design pattern provides a clean mechanism to cater common design problems using generic guidelines.

Big data design patterns provide the concrete representation of analysis and technology centric patterns of most common occurring problems in big data environment [75]. These design patterns provide the building blocks for the efficient design of big data architecture. The standardization and integration of design patterns can be seen as the potential candidates for the efficient and effective convergence of HPC and big data. Figure 23.6 shows the logical architecture of different layers and design patterns (HPC & Big Data) can then be applied at distinct levels to address the issues related to big data and HPC convergence. One of the challenges associated with data visualization and interactive management is huge volume, variety, and velocity of data and is often hard to evaluate and reapply the design solution. The visualization and management layer involves applying patterns for distributed and parallel visualization, interactive data exploration, rendering data visualization, real-time monitoring for live analysis and recommendations.

The analytics/processing layer includes patterns for analytics and depending on the problem domain includes in situ, in-transit, real-time, or batch processing. Advanced analytics requires predictions, advance algorithms, simulations, and real-time decisions that require high performance computing for processing and managing massive volume of data [76].

There is a trade-off between performance, resilience, and power consumption. Trade-off patterns need to identify and accommodate these trade-offs in best possible way by indulging the best practices from both HPC and big data communities. The processing pattern includes analytics patterns for unstructured and structured data, algorithms for conversion of unstructured to structured data, large-scale batch and graph based processing patterns, and also parallel design patterns. The access/storage layer includes design patterns for the effective and efficient retrieval and storage mechanism for parallel and distributed file systems. This includes data size reduction for high volume hierarchical, linked, tabular and binary cognitive storage for real-time in-direct and integrated access. The cognitive storage with

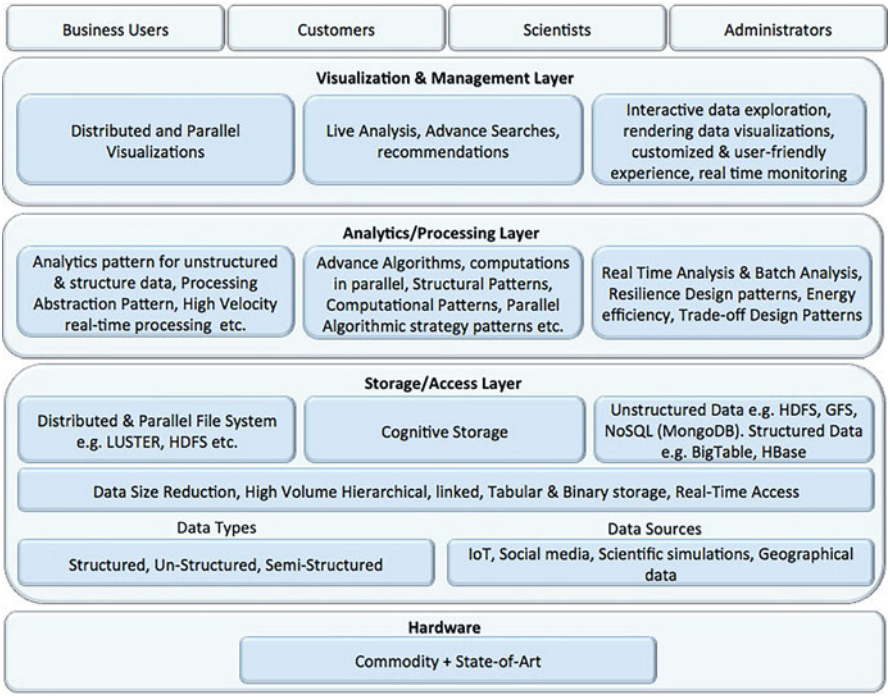


Fig. 23.6 Design Patterns Based Converged Future

learning ability is to automate the process of data purging by keeping only relevant and important data for cost effective storage and improved performance.

HPC software development community lack the expertise of software engineering principles as these patterns define the building blocks of software engineering and are fundamental to architect parallel software. There is a need to invest the research efforts towards exploration of innovative approaches to make use of design patterns and skeletons to overcome scalability, elasticity, adaptability, robustness, storage, parallelization, and other processing challenges of the unified HPC and big data environment.

23.9 Conclusion

The increased processing power, emergence of big data resources, and real-time analytical solutions are the prime drivers that push the realm of big data. HPC and big data systems are different not only at programming level but also have different architecture. The challenges associated with inevitable integration of HPC and big data are immense and solutions are starting to emerge at distinct levels

of ecosystem. As we are heading towards convergence of both, we will have to deal with modality, complexity, and vast amount of data. Currently we have distinct and perhaps overlapping set of design choices at various levels of infrastructure. A single system architecture but with enough configurability in it that you can actually serve different design points between compute intensive and design intensive. The single system architecture overcomes the cost and complexity of moving data. It also improves the total cost of ownership and brings in flexibility to manage workflows and maximize system utilization. Realizing these benefits requires coordinated design efforts around key elements of the system, i.e., compute (multicore, FPGA), interconnect (next generation fabric), memory (Non Volatile memory, storage burst buffer, Lustre file system). This coordinated effort may result in useable, effective, and scalable software infrastructure.

The connected and ubiquitous synergy between HPC and big data is expected to deliver the results which cannot be achieved by either alone. There is a need for the leading enterprises to use HPC technology to explore efficiently huge volume of heterogeneous data to surpass static searches into dynamic pattern discovery for the competitive advantage. The integration of computing power in HPC and demands for a quick and real-time analytics for big data with cognitive technology (computer vision techniques, machine learning, natural language processing) are considered as reshaping the future technology for accelerating analytics and deriving meaningful insights for efficient decision-making.

Acknowledgments The authors acknowledge with thanks the technical and financial support from the Deanship of Scientific Research (DSR) at the King Abdul-Aziz University (KAU), Jeddah, Saudi Arabia, under the grant number G-673-793-38. The work carried out in this paper is supported by the HPC Center at the King Abdul-Aziz University.

References

1. Singh, K., Kaur, R.: Hadoop: addressing challenges of big data. In: 2014 IEEE International Advance Computing Conference (IACC), pp. 686–689. IEEE (2014)
2. Mehmood, R., Faisal, M.A., Altowaijri, S.: Future networked healthcare systems: a review and case study. In: Boucadair, M., Jacquenet, C. (eds.) Handbook of research on redesigning the future of internet architectures, pp. 531–558. IGI Global, Hershey (2015)
3. Charl, S.: IBM—HPC and HPDA for the cognitive journey with OpenPOWER. https://www-03.ibm.com/systems/power/solutions/bigdata-analytics/smartpaper/high-value-insights.html
4. Alzahrani, S., Ikbal, M.R., Mehmood, R., Fayez, M., Katib, I.: Performance evaluation of Jacobi iterative solution for sparse linear equation system on multicore and manycore architectures. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 296–305. Springer, Cham (2018)
5. Alyahya, H., Mehmood, R., Katib, I.: Parallel sparse matrix vector multiplication on Intel MIC: performance analysis. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 306–322. Springer, Cham (2018)

6. Kwiatkowska, M., Mehmood, R.: Out-of-core solution of large linear systems of equations arising from stochastic modelling. In: Hermanns, H., S.R. (eds.) Process Algebra and Probabilistic Methods: Performance Modeling and Verification. PAPM-PROBMIV, pp. 135–151. Springer, Berlin (2002)
7. Mehmood, R.: Disk-based techniques for efficient solution of large Markov chains. http://www.academia.edu/download/3361709/rashidsthesis.pdf (2004)
8. Mehmood, R., Crowcroft, J.: Parallel iterative solution method for large sparse linear equation systems. Technical Report Number UCAM-CL-TR-650, Computer Laboratory, University of Cambridge, Cambridge, UK (2005)
9. Kwiatkowska, M., Mehmood, R., Norman, G., Parker, D.: A symbolic out-of-core solution method for Markov Models. Electron. Notes Theor. Comput. Sci. **68**, 589–604 (2002)
10. Mehmood, R., Lu, J.A.: Computational Markovian analysis of large systems. J. Manuf. Technol. Manag. **22**, 804–817 (2011)
11. Kwiatkowska, M., Parker, D., Yi Zhang, Y., Mehmood, R.: Dual-processor parallelisation of symbolic probabilistic model checking. In: The IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004 (MASCOTS 2004). Proceedings, pp. 123–130. IEEE (2004)
12. Mehmood, R.: Serial disk-based analysis of large stochastic models. In: Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.P., S.M. (eds.) Validation of Stochastic Systems, pp. 230–255. Springer, Berlin (2004)
13. Mehmood, R., Crowcroft, J., Elmirghani, J.M.H.: A parallel implicit method for the steady-state solution of CTMCs. In: 14th IEEE International Symposium on Modeling, Analysis, and Simulation, pp. 293–302. IEEE (2006)
14. Keable, C: The convergence of high performance computing and big data—ascent. https://ascent.atos.net/convergence-high-performance-computing-big-data/
15. Usman, S., Mehmood, R., Katib, I.: Big data and HPC convergence: The cutting edge and outlook. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 11–26. Springer, Cham (2018)
16. Padhy, R.P.: Big data processing with Hadoop-MapReduce in cloud systems. IJ-CLOSER Int. J. Cloud Comput. Serv. Sci. **2**, 233–245 (2012)
17. Hess, K.: Hadoop vs. spark: the new age of big data. http://www.datamation.com/data-center/hadoop-vs.-spark-the-new-age-of-big-data.html
18. Muhammad, J.: Is Apache Spark going to replace Hadoop? http://aptuz.com/blog/is-apache-spark-going-to-replace-hadoop/
19. OLCF staff writer: OLCF group to offer spark on-demand data analysis. https://www.olcf.ornl.gov/2016/03/29/olcf-group-to-offer-spark-on-demand-data-analysis/
20. Jost, G., Jin, H.-Q., anMey, D., Hatay, F.F.: Comparing the OpenMP, MPI, and hybrid programming paradigm on an SMP cluster. In: European Workshop on OpenMP and Applications. Aachen Germany (2003)
21. De Wael, M., Marr, S., De Fraine, B., Van Cutsem, T., De Meuter, W.: Partitioned global address space languages. ACM Comput. Surv. **47**, 1–27 (2015)
22. Calin, G., Derevenetc, E., Majumdar, R., Meyer, R.: A theory of partitioned global address spaces (2013)
23. Joseph, E., Sorensen, B.: IDC update on how big data is redefining high performance computing. https://www.tacc.utexas.edu/documents/1084364/1136739/IDC+HPDA+Briefing+slides+10.21.2014_2.pdf
24. Geist, A., Lucas, R.: Whitepaper on the Major Computer Science Challenges at Exascale (2009)
25. Krishnan, S., Tatineni, M., Baru, C.: myHadoop-Hadoop-on-demand on traditional HPC resources (2011)

26. Xuan, P., Denton, J., Ge, R., Srimani, P.K., Luo, F.: Big data analytics on traditional HPC infrastructure using two-level storage (2015)
27. Is Hadoop the New HPC. http://www.admin-magazine.com/HPC/Articles/Is-Hadoop-the-New-HPC
28. Katal, A., Wazid, M., Goudar, R.H.: Big data: issues, challenges, tools and good practices. In: 2013 Sixth International Conference on Contemporary Computing (IC3), pp. 404–409. IEEE (2013)
29. Philip Chen, C.L., Zhang, C.Y.: Data-intensive applications, challenges, techniques and technologies: a survey on big data. Inf. Sci. (Ny). **275**, 314–347 (2014)
30. Chen, M., Mao, S., Liu, Y.: Big data: a survey. Mob. Netw. Appl. **19**, 171–209 (2014)
31. Xu, Z., Shi, Y.: Exploring big data analysis: fundamental scientific problems. Ann. Data Sci. **2**(4), 363–372 (2015)
32. Hashem, I.A.T., Yaqoob, I., Badrul Anuar, N., Mokhtar, S., Gani, A., Ullah Khan, S.: The rise of "Big Data" on cloud computing: review and open research issues. Inf. Syst. **47**, 98–115 (2014)
33. Reed, D.A., Dongarra, J.: Exascale computing and big data. Commun. ACM. **58**, 56–68 (2015)
34. Asch, M., Moore, T., Badia, R., Beck, M., Beckman, P., Bidot, T., Bodin, F., Cappello, F., Choudhary, A., de Supinski, B., Deelman, E., Dongarra, J., Dubey, A., Fox, G., Fu, H., Girona, S., Gropp, W., Heroux, M., Ishikawa, Y., Keahey, K., Keyes, D., Kramer, W., Lavignon, J.-F., Lu, Y., Matsuoka, S., Mohr, B., Reed, D., Requena, S., Saltz, J., Schulthess, T., Stevens, R., Swany, M., Szalay, A., Tang, W., Varoquaux, G., Vilotte, J.-P., Wisniewski, R., Xu, Z., Zacharov, I.: Big data and extreme-scale computing. Int. J. High Perform. Comput. Appl. **32**, 435–479 (2018)
35. Islam, N.S., Lu, X., Wasi-ur-Rahman, M., Shankar, D., Panda, D.K.: Triple-H: a hybrid approach to accelerate HDFS on HPC clusters with heterogeneous storage architecture. In: 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 101–110. IEEE (2015)
36. Ranger, C., Raghuraman, R., Penmetsa, A., Bradski, G., Kozyrakis, C.: Evaluating MapReduce for multi-core and multiprocessor systems. In: 2007 IEEE 13th International Symposium on High Performance Computer Architecture, pp. 13–24. IEEE (2007)
37. Schwan, P., Schwan, P.: Lustre: building a file system for 1000-node clusters. In: Proceedings of 2003 LINUX Symposium (2003)
38. Ghemawat, S., Gobioff, H., Leung, S.-T., Ghemawat, S., Gobioff, H., Leung, S.-T.: The Google file system. In: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles—SOSP'03, p. 29. ACM Press, New York (2003)
39. Owre, S., Shankar, N., Rushby, J.M., Stringer-Calvert, D.W.J.: PVS System Guide (2001)
40. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The Hadoop Distributed File System. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), pp. 1–10. IEEE (2010)
41. Apache Hadoop 2.9.0—C API libhdfs. https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/LibHdfs.html
42. Martinec, J., Rango, A., Major, E.: The Snowmelt-Runoff Model (SRM) user's manual (1983)
43. Rajasekar, A., Moore, R., Hou, C.-Y., Lee, C.A., Marciano, R., de Torcy, A., Wan, M., Schroeder, W., Chen, S.-Y., Gilbert, L., Tooby, P., Zhu, B.: iRODS primer: integrated rule-oriented data system. In: Synth. Lect. Inf. Concepts, Retrieval, Serv. 2, 1–143 (2010)
44. Plimpton, S.J., Devine, K.D.: MapReduce in MPI for large-scale graph algorithms. Parallel Comput. **37**, 610–632 (2011)
45. Mantha, P.K., Luckow, A., Jha, S.: Pilot-MapReduce. In: Proceedings of Third International Workshop on MapReduce and Its Applications Date—MapReduce'12, p. 17. ACM Press, New York (2012)
46. Tiwari, N., Sarkar, S., Bellur, U., Indrawan, M.: An empirical study of Hadoop's energy efficiency on a HPC cluster. Procedia Comput. Sci. **29**, 62–72 (2014)
47. Woodie, A: Does InfiniBand have a future on Hadoop? http://www.datanami.com/2015/08/04/does-infiniband-have-a-future-on-hadoop/

48. Veiga, J., Exp, R.R., Taboada, G.L., Touri, J.: Analysis and evaluation of big data computing solutions in an HPC environment (2015)
49. Wang, Y., Jiao, Y., Xu, C., Li, X., Wang, T., Que, X., Cira, C., Wang, B., Liu, Z., Bailey, B., Yu, W.: Assessing the performance impact of high-speed interconnects on MapReduce. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) LNCS, vol. 8163, pp. 148–163 (2014)
50. Islam, N.S., Lu, X., Wasi-ur-Rahman, M., Panda, D.K.: Can parallel replication benefit Hadoop distributed file system for high performance interconnects? In: 2013 IEEE 21st Annual Symposium on High-Performance Interconnects, pp. 75–78. IEEE (2013)
51. Moore, J., Chase, J., Ranganathan, P., Sharma, R.: Making scheduling cool: temperature-aware workload placement in data centers (2005)
52. Rajovic, N., Puzovic, N., Vilanova, L., Villavieja, C., Ramirez, A.: The low-power architecture approach towards exascale computing. In: Proceedings of the Second Workshop on Scalable Algorithms for Large-Scale Systems—ScalA'11, p. 1. ACM Press, New York (2011)
53. Cappello, F.: Fault tolerance in petascale/exascale systems: current knowledge, challenges and research opportunities. Int. J. High Perform. Comput. Appl. **23**, 212–226 (2009)
54. Gutierrez, D: The convergence of big data and HPC—inside BIGDATA. https://insidebigdata.com/2016/10/25/the-convergence-of-big-data-and-hpc/
55. High performance data analytics (HPDA) market-forecast 2022. https://www.marketresearchfuture.com/reports/high-performance-data-analytics-hpda-market
56. Willard, C.G., Snell, A., Segervall, L.: Top six predictions for HPC in 2015 (2015)
57. Egham: Gartner says 8.4 billion connected "things" will be in use in 2017, up 31 percent from 2016. http://www.gartner.com/newsroom/id/3598917
58. Ahmed, W., Khan, M., Khan, A.A., Mehmood, R., Algarni, A., Albeshri, A., Katib, I.: A framework for faster porting of scientific applications between heterogeneous clouds. In: Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, pp. 27–43. Springer, Cham (2018)
59. Alotaibi, S., Mehmood, R.: Big data enabled healthcare supply chain management: opportunities and challenges. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 207–215. Springer, Cham (2018)
60. Alamoudi, E., Mehmood, R., Albeshri, A., Gojobori, T.: DNA profiling methods and tools: a review. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 216–231. Springer, Cham (2018)
61. Al Shehri, W., Mehmood, R., Alayyaf, H.: A smart pain management system using big data computing. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 232–246. Springer, Cham (2018)
62. Khanum, A., Alvi, A., Mehmood, R.: Towards a semantically enriched computational intelligence (SECI) framework for smart farming. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 247–257. Springer, Cham (2018)
63. Suma, S., Mehmood, R., Albeshri, A.: Automatic event detection in smart cities using big data analytics. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 111–122. Springer, Cham (2018)

64. Aqib, M., Mehmood, R., Albeshri, A., Alzahrani, A.: Disaster management in smart cities by forecasting traffic plan using deep learning and GPUs. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 139–154. Springer, Cham (2018)
65. Alam, F., Mehmood, R., Katib, I.: D2TFRS: an object recognition method for autonomous vehicles based on RGB and spatial values of pixels. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 155–168. Springer, Cham (2018)
66. Muhammed, T., Mehmood, R., Albeshri, A.: Enabling reliable and resilient IoT based smart city applications. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 169–184. Springer, Cham (2018)
67. Al-Dhubhani, R., Mehmood, R., Katib, I., Algarni, A.: Location privacy in smart cities era. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 123–138. Springer, Cham (2018)
68. Alomari, E., Mehmood, R.: Analysis of Tweets in Arabic Language for detection of road traffic conditions. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 98–110. Springer, Cham (2018)
69. Arfat, Y., Aqib, M., Mehmood, R., Albeshri, A., Katib, I., Albogami, N., Alzahrani, A.: Enabling smarter societies through mobile big data fogs and clouds. Procedia Comput. Sci. **109**, 1128–1133 (2017)
70. Schlingensiepen, J., Nemtanu, F., Mehmood, R., McCluskey, L.: Autonomic transport management systems—enabler for smart cities, personalized medicine, participation and industry grid/industry 4.0. In: Intelligent Transportation Systems—Problems and Perspectives, pp. 3–35. Springer, Cham (2016)
71. Arfat, Y., Mehmood, R., Albeshri, A.: Parallel shortest path graph computations of United States Road Network Data on Apache Spark. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. SCITA 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 323–336. Springer, Cham (2018)
72. El Baz, D.: IoT and the need for high performance computing. In: 2014 International Conference on Identification, Information and Knowledge in the Internet of Things, pp. 1–6. IEEE (2014)
73. Conway, S: High performance data analysis (HPDA): HPC—big data convergence—insideHPC (2017)
74. Keutzer, K., Tim, M.: Our pattern language_our pattern language, file:///Users/abdulmanan/Desktop/Our Pattern Language_Our Pattern Language.htm (2016)
75. Bodkin, R., Bodkin, R.: Big data patterns, pp. 1–23 (2017)
76. Mysore, D., Khupat, S., Jain, S.: Big data architecture and patterns, Part 1: introduction to big data classification and architecture. https://www.ibm.com/developerworks/library/bd-archpatterns1/index.html

Part V
Internet of Things (IoT)

Chapter 24
Towards a Runtime Testing Framework for Dynamically Adaptable Internet of Things Networks in Smart Cities

Moez Krichen and Mariam Lahami

24.1 Introduction

Recent advances in communication and sensing devices make our everyday objects smarter. This smartness is resulted from the capability of objects to sense the environment, to process the captured (sensed) data, and to communicate it to users either directly or through the internet. Taking an example of the object "lamp," a classical lamp needs to be wired, linked to the electricity in order to produce light and it does not handle more than the on and off states. This lamp becomes smarter if it is equipped with sensors that can detect environment luminosity and adjust its brightness automatically based on the sensed value. Moreover, this lamp can be equipped with a communication system and therefore can be remotely controlled and supervised (e.g., energy consumption). The IoT refers to the ability of everyday objects to connect to the Internet and to send and receive data. The integration of these smart objects to the Internet infrastructure is promoting a new generation of innovative and valuable services for people. These services include home automation, traffic control, public transportation, smart water metering, waste and energy management, etc. When integrated in a city context, they make citizens live better and so form the modern smart city.

M. Krichen (✉)
Faculty of CSIT, Al-Baha University, Al Baha, Saudi Arabia

ReDCAD Laboratory, University of Sfax, Sfax, Tunisia
e-mail: moez.krichen@redcad.org

M. Lahami
ReDCAD Laboratory, University of Sfax, Sfax, Tunisia
e-mail: mariam.lahami@redcad.org

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_24

In recent years, several research works are shaping the smart cities concept [9]. In October 2015, ITU-T's Focus Group on Smart Sustainable Cities (FG-SSC) agreed on the following definition of a smart sustainable city: "A Smart Sustainable City (SSC) is an innovative city that uses information and communication technologies (ICTs) and other means to improve quality of life, efficiency of urban operation and services, and competitiveness, while ensuring that it meets the needs of present and future generations with respect to economic, social and environmental aspects." Based on this definition, the main goal for SSC is to enhance the quality of life of its citizens across multiple, interrelated dimensions, including the provision and access to water resources, energy, transportation and mobility, education, environment, waste management, housing, and livelihoods (e.g., jobs), utilizing ICTs as the key medium. Therefore, the IoT as a promising ICT technology will play a major role in the development of these new smart cities. With IoT, objects like phones, cars, household appliances, or clothes become wirelessly connected and can sense and share data.

Nowadays, distributed component-based systems tend to evolve dynamically without stopping their execution. Known as *dynamically adaptable and distributed systems*, these systems are currently playing an important role in society's services. Indeed, the growing demand for such systems is obvious in several application domains such as crisis management [10] and medical monitoring [4]. This demand is stressed by the complex, mobile, and critical nature of these applications that also need to continue meeting their functional and non-functional requirements and to support advanced properties. Nevertheless, dynamic adaptations of component-based systems may generate new risks of bugs, unpredicted interactions, unintended operation modes, and performance degradation. This may cause system malfunctions and guide its execution to an unsafe state. Therefore, guaranteeing their high quality and their trustworthiness remains a crucial requirement to be considered. One of the most promising ways of testing dynamic systems is the use of an emerging technique, called *runtime testing*.

In this work, we propose a standard-based test execution platform for dynamically adaptable IoT networks in smart cities which affords a platform-independent test system for isolating and executing runtime tests. This platform uses the TTCN3 standard and considers both structural and behavioral adaptations. Moreover, our platform is equipped with a test isolation layer that reduces the risk of interference between testing processes and business processes. We also compute a minimal subset of test cases to run and efficiently distribute them among the execution nodes. The main research contributions presented in this paper are the following. The minimal subset of test cases is obtained using a smart generation algorithm which keeps old tests cases which are still valid and replaces invalid ones by new generated or updated test cases.

The remainder of this paper is organized as follows. In Sect. 24.2, we provide techniques for runtime testing for structural adaptations. Section 24.3 is dedicated to runtime testing of behavioral adaptations. Finally, Sect. 24.4 provides a conclusion that summarizes the paper and discusses items for future work.

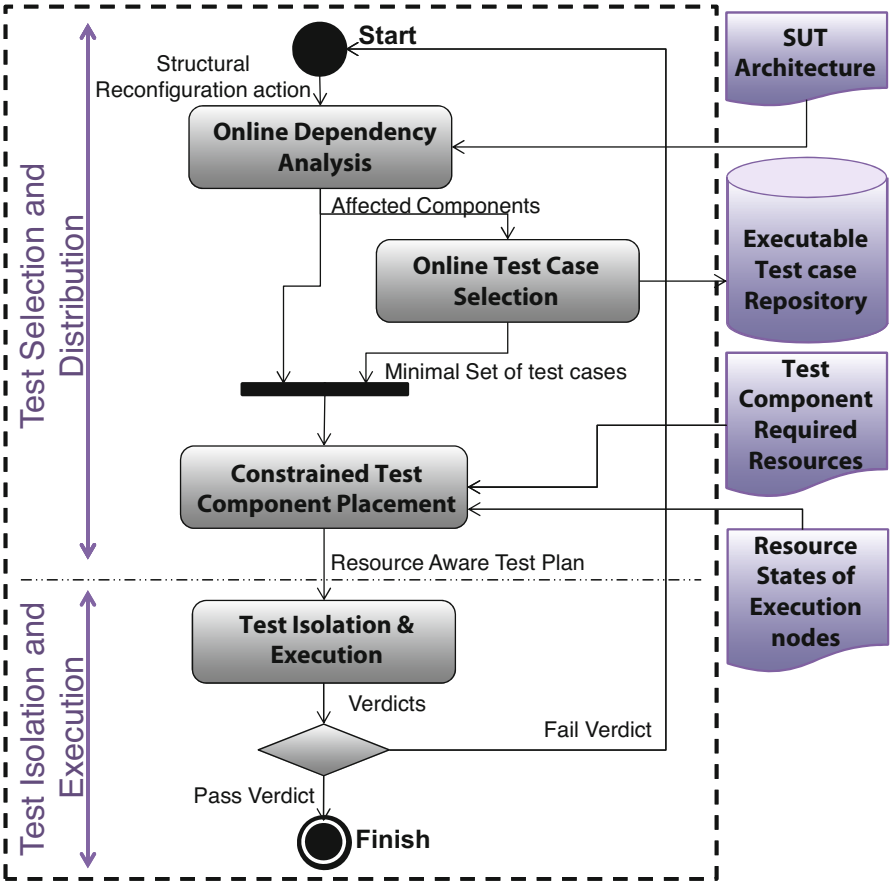


Fig. 24.1 Runtime testing process for the validation of structural adaptations

24.2 Runtime Testing for Structural Adaptations

Testing at design-time or even at deployment-time usually demonstrates that the system under test, SUT, satisfies its functional and non-functional requirements. However, its applicability becomes limited and irrelevant when this system is adapted at runtime according to evolving requirements and environmental conditions that were not explicitly specified at design-time. For this reason, runtime testing is strongly required to extend assurance from design-time to runtime.

The process depicted in Fig. 24.1 spans the different steps to fulfill with the aim of executing runtime tests when structural reconfiguration actions are triggered, as follows:

– **Online dependency analysis:** In this step, we focus on identifying the affected components and compositions by a structural reconfiguration action.

- **Online test case selection:** Once the affected parts of the system are identified, we look for their corresponding test cases that are stored in the *executable test case repository*.
- **Constrained test component placement:** Test components are assigned to execution nodes in an appropriate manner with respect to resource and connectivity constraints.
- **Test isolation and execution:** A test isolation layer is set up then test components are dynamically created and test cases are executed.

More details are presented in the next sections.

24.2.1 Online Dependency Analysis

To reduce the time cost and the resource burden of the runtime testing process, the key idea is to avoid the re-execution of all tests at runtime when structural adaptations occur. Thus, we use the dependency analysis approach with the aim of determining the parts of the system impacted by dynamic evolutions and then computing a minimal set of tests to rerun. In fact, the dependency analysis technique is widely used in various software engineering activities including testing [15], maintenance, and evolution [16].

Definition Dependencies between components are defined in [1] as "the reliance of a component on other(s) to support a specific functionality." It is also considered as a binary relation between two components. A component A is an antecedent to another component B if its data or functionalities are utilized by B. Equivalently, A component B is dependent on another component A if it utilizes data or functionalities of A. Formally, the relation $\rightarrow$ called "*Depends on*" is defined in [13] where $B \rightarrow A$ means that the component B depends on the component A. The set of all dependencies in a component-based system is defined as: $\mathcal{D} = \{(C_i, C_j) : C_i, C_j \in \mathcal{S} \wedge C_i \rightarrow C_j\}$ where $\mathcal{S}$ is the set of components in the system. Accordingly, the current system configuration is a set of components and its dependencies $Con = (\mathcal{S}, \mathcal{D})$.

Several forms of dependencies component-based systems are identified in the literature [15]. For instance, we mention data dependency (i.e., data defined in one component is used in another component), control dependency (i.e., caused by sending a message from one component to another component), etc. The main dependency form that we support in this work is the interface dependency, which means that a component requires (respectively provides) a service from (respectively to) another component.

Dependency Representation To represent and analyze component dependencies, two formalisms are generally described: a *component dependency graph* (CDG) and a *component dependency matrix* (CDM). A CDG is a directed graph denoted by $\mathcal{G} = (\mathcal{S}, \mathcal{D})$ where $\mathcal{S}$ is a finite nonempty set of vertices representing system's

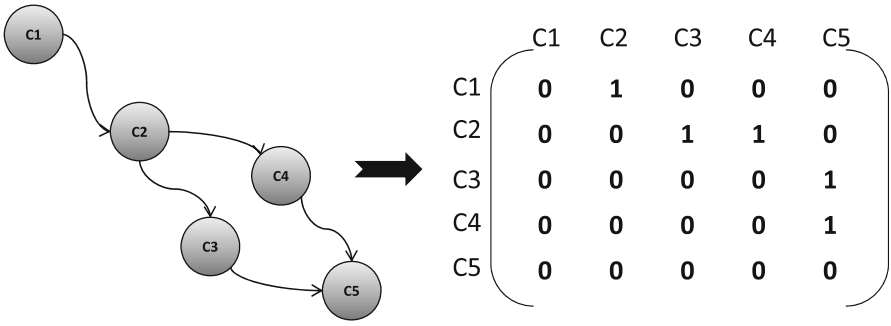


Fig. 24.2 A CDG and its CDM representing direct dependencies

components and $\mathcal{D}$ is a set of edges between two vertices, $\mathcal{D} \subseteq (\mathcal{S} \times \mathcal{S})$. A CDM is defined as a 0-1 adjacency matrix $\mathcal{AM}_{n \times n}$ that represents direct dependencies in a component-based system. In this matrix, each component is represented by a column and a row. If a component C_i depends on a component C_j, then $d_{ij} = 1$ otherwise $d_{ij} = 0$. Figure 24.2 shows an example of dependency graph and its corresponding adjacency matrix.

Initially, $\mathcal{D}$ represents only direct dependencies between components. In order to gather all indirect dependencies in the component-based system, the transitive closure of the graph has to be calculated. Several transitive closure algorithms have been widely studied in the literature such as the Roy–Warshall algorithm and its modification proposed by Warren [8].

24.2.2 Online Test Case Selection

This concern has been extensively studied in the literature. In fact, various regression test selection techniques have been proposed with the purpose of identifying a subset of valid test cases from an initial test suite that tests the affected parts of a program. These techniques usually select regression tests based on data and control dependency analysis [17].

Two kinds of tests are considered after the occurrence of dynamic adaptations. On the one hand, unit tests are executed to validate individual affected components. On the other hand, integration tests are performed to check interactions and interoperability between components.

Let us take an example with four components and a dependency graph that looks like Fig. 24.3. Assume that C_2 is replaced with a new version. Thus, two dependence paths are identified: $C_1 \rightarrow C_2 \rightarrow C_3$ and $C_1 \rightarrow C_2 \rightarrow C_4$. As a result, the mapping to integration tests produces: $ITC1C2C3$ and $ITC1C2C4$ have to be rerun.

Recall that tests are written in the TTCN-3 notation and are executed by TTCN-3 test components. As depicted in Fig. 24.4a, an MTC component is only charged with

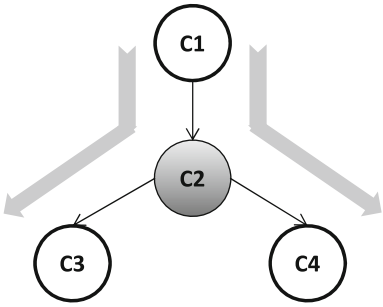


Fig. 24.3 Illustrative example of dependence path computation

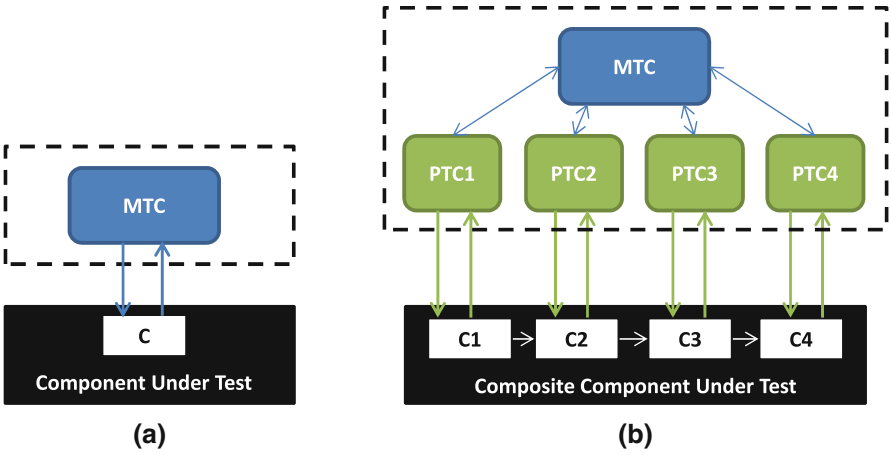


Fig. 24.4 TTCN-3 test configuration for unit and integration testing. (**a**) Unit test configuration. (**b**) Integration test configuration

executing a unit test. It shares this responsibility with other PTC components when an integration test is executed (see Fig. 24.4b). Each PTC is created to simulate a test call from a component to another at lower hierarchy in the dependence path. The following subsection copes with test case distribution and more precisely with main test components assignment to execution nodes.

24.2.3 Constrained Test Component Placement

In the following subsections, we discuss how to formalize resource and connectivity constraints and how to find the adequate deployment host for each test involved in the runtime testing process.

Resource Allocation Issue For each node in the execution environment, three resources are monitored during the SUT execution: the available memory, the current CPU load, and the battery level. The value of each resource can be directly

captured on each node through the use of internal monitors. These values are measured after the runtime reconfiguration and before starting the testing activity. Formally, provided resources of m execution nodes are represented through three vectors: C contains the CPU load, R provides the available RAM, and B introduces the battery level.

$$C = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{pmatrix} \quad R = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{pmatrix} \quad B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

The resources required by the n test components are initially computed at the deployment-time after a preliminary test run. Similarly, they are formalized over three vectors: D_c that contains the required CPU, D_r that introduces the required RAM, and D_b that contains the required battery by each test.

$$D_c = \begin{pmatrix} dc_1 \\ dc_2 \\ \vdots \\ dc_n \end{pmatrix} \quad D_r = \begin{pmatrix} dr_1 \\ dr_2 \\ \vdots \\ dr_n \end{pmatrix} \quad D_b = \begin{pmatrix} db_1 \\ db_2 \\ \vdots \\ db_n \end{pmatrix}$$

As the proposed framework is resource aware, the overall resources required by n test components must not exceed the available resources in m nodes. This rule is formalized as follows:

$$\begin{cases} \sum_{i=1}^{n} x_{ij} dc_i \leq c_j & \forall j \in \{1, \ldots, m\} \\ \sum_{i=1}^{n} x_{ij} dr_i \leq r_j & \forall j \in \{1, \ldots, m\} \\ \sum_{i=1}^{n} x_{ij} db_i \leq b_j & \forall j \in \{1, \ldots, m\} \end{cases} \tag{24.1}$$

where the two dimensional variable x_{ij} can be equal to 1 if the corresponding test component i is assigned to the node j, 0 otherwise.

Connectivity Issue Dynamic environments are characterized by frequent and unpredictable changes in connectivity caused by firewalls, non-routing networks, node mobility, etc. For this reason, we have to pay attention when assigning a test component to a host computer by finding at least one route in the network to communicate with the component under test. For each test component, a set of forbidden nodes to discard during the constrained test component placement step is defined. This connectivity constraint is denoted as follows:

$$x_{ij} = 0 \quad \forall j \in forbiddenNodeSet(i) \tag{24.2}$$

Finding a satisfying test placement solution is achieved by fitting the former constraints (24.1) and (24.2). The latter can be seen as a *constraint satisfaction problem* (CSP) [5].

Optimizing the Test Component Placement Problem Looking for an optimal test placement solution consists in identifying the best node to host the concerned test component in response with two criteria: its distance from the node under test and its link bandwidth capacity. To do so, we are asked to attribute a profit value p_{ij} for assigning the test component i to a node j. For this aim, a matrix $\mathcal{P}_{n\times m}$ is computed as follows:

$$p_{ij} = \begin{cases} 0 & \text{if} \quad j \in forbiddenNodeSet(i) \\ maxP - k \times step_p & \text{otherwise} \end{cases} \tag{24.3}$$

where $maxP$ is a constant, $step_p = \frac{maxP}{m}$, k corresponds to the index of a node j in a *rank vector* that is computed for each node under test. This vector corresponds to a classification of the connected nodes according to two criteria: the distance from the testing node to the node under test [12] and the link bandwidth capacities.

As a result, the constrained test component placement module generates the best deployment host for each test component involved in the runtime testing process by maximizing the total profit value while fitting the former resource and connectivity constraints. Thus, this problem is formalized as a variant of the knapsack problem, called *multiple multidimensional knapsack problem* (MMKP).

$$\text{MMKP} = \begin{cases} \text{maximize} \quad \mathcal{Z} = \sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij} x_{ij} & (24.4) \\ \text{subject to} \quad (24.1) \quad \text{and} \quad (24.2) & \\ \sum_{j=1}^{m} x_{ij} = 1 \quad \forall i \in \{1, \ldots, n\} & (24.5) \\ x_{ij} \in \{0, 1\} \quad \forall i \in \{1, \ldots, n\} \quad \text{and} \quad \forall j \in \{1, \ldots, m\} & \end{cases}$$

Constraint (24.4) corresponds to the objective function that maximizes test component profits while satisfying resource (24.1) and connectivity (24.2) constraints. Constraint (24.5) indicates that each test component has to be assigned to at most one node.

24.2.4 Test Isolation and Execution Support

With the purpose of alleviating the complexity of testing adaptable and distributed systems, we propose a test system called, ***TT****CN-3 test system for* ***R****untime* ***T****esting* (TT4RT) [11].

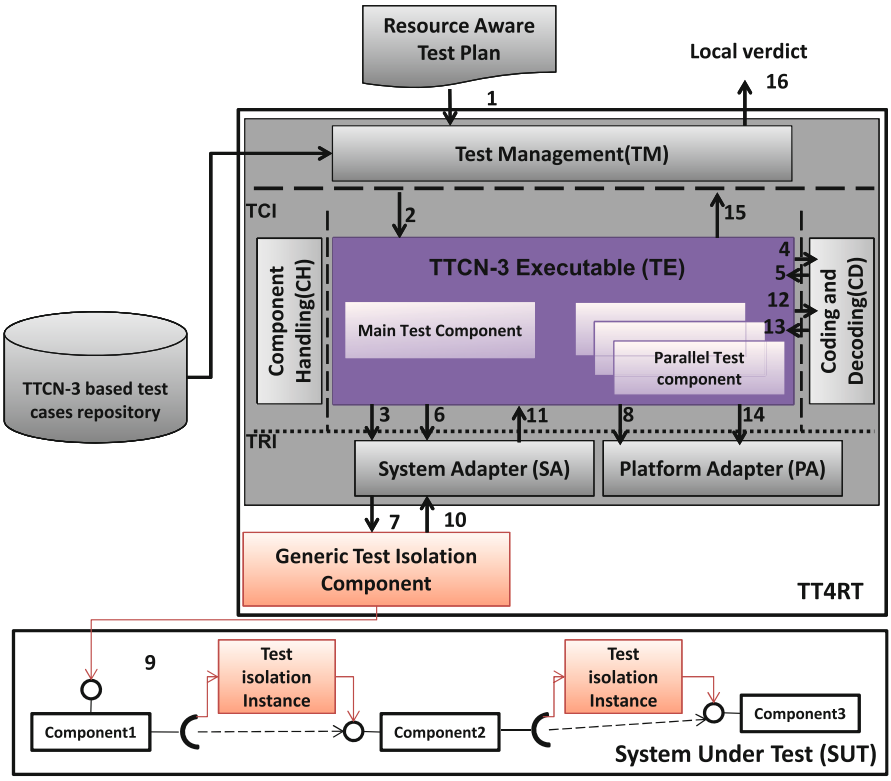


Fig. 24.5 Internal interactions in the TT4RT system

Detailed Interactions of TT4RT Components TT4RT relies on the classical TTCN-3 test system. Thus, it reuses all its constituents, namely test management (TM), TTCN-3 executable (TE), component handling (CH), coding and decoding (CD), system adapter (SA), and platform adapter (PA). These entities are briefly introduced below. As depicted in Fig. 24.5, a new *generic test isolation component* is added to the TTCN-3 reference architecture with the aim of handling test isolation concerns. The next steps define the different components of TT4RT and their internal interactions:

- When a reconfiguration action is triggered, the RATP (resource aware test plan) file is generated and it is considered as an input to the TT4RT test system (Step **1**).
- The test execution is initiated by the TM entity which is charged with starting and stopping runtime tests (Step **2**).
- Once the test process is started, the TE entity (i.e., which is responsible for executing the compiled TTCN-3 code) creates the involved test components and informs the SA entity (i.e., which is charged with propagating test requests from

TE to SUT) with this start-up in order to set up its communication facilities (Step **3**).
- Next, TE invokes the CD entity in order to encode the test data from a structured TTCN-3 value into a form that will be accepted by the SUT (Step **4**).
- The encoded test data is passed back to the TE entity as a binary string and forwarded to the SUT via the SA entity (Steps **5–7**).
- After the test data is sent, a timer can be started (Step **8**).
- The generic test isolation component, implementing test isolation facilities, intercepts the test request, identifies the component under test and its supported test isolation technique, and prepares the test environment (Steps **7–9**).
- Different test isolation instances are automatically created to perform test isolation inter-component invocations (Step **9**).
- The SUT response is forwarded to the SA entity through the generic test isolation component. The given response is an encoded value that has to be decoded in order to be understandable by the TTCN-3 test system (Step **10**).
- For this purpose, the SA entity forwards the encoded test data to the TE entity (Step **11**).
- The TE entity transmits the encoded response to the CD entity with the intention of decoding it into a structured TTCN-3 value (Step **12**).
- The decoded response is passed back to the TE that stops the running timer and finally computes a verdict (pass, fail, or inconclusive) for the current test case (Steps **13–15**).
- Finally, a local verdict is computed depending on the obtained verdicts for test cases executed by the current TT4RT instance (Step **16**).

Overview of the Generic Test Isolation Component As outlined in Fig. 24.6, the proposed policy is executed while a test request is intercepted from the system adapter entity. Five strategies can be applied in response to the testability degree of a component under test (CUT). With the assumption that the CUT is testable, the test request can be redirected to one or more test operations provided by its corresponding test interface or its associated aspect (particularly in the advice part) when the aspect-based technique is used. If the component under test is test aware, the tagging technique is applied and the CUT is invoked by tagging the input test data with a flag to discriminate them from business data. If we deal with untestable components, either cloning or blocking techniques can be performed. For a test sensitive component, a clone is created and the test request is redirected to it. Regarding the blocking strategy, it consists in interrupting the activity of the component under test consumers for a lapse of time that corresponds to the test duration. During this period, all business requests are delayed until the end of the test. Once the test is achieved, the component under test consumers are unlocked and the delayed requests are treated.

The Adopted Distributed Architecture The TTCN-3 standard offers concepts related to test configurations, test components, their communicating ports between each other and with the SUT, their execution, and their termination only at an

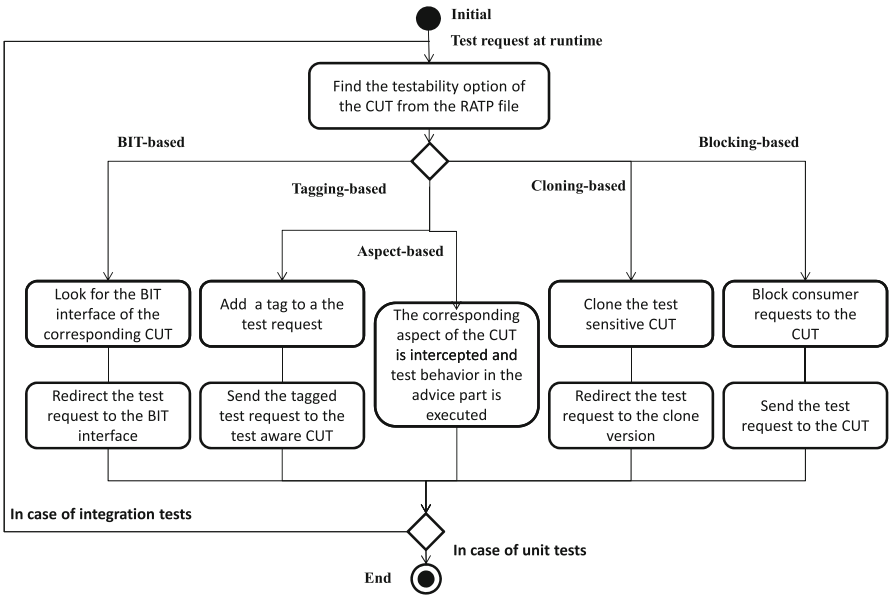


Fig. 24.6 Test isolation policy

abstract level. Nevertheless, the means to control the distributed execution of these test components are not explicitly defined in the current specification. Regarding this issue, we propose our own test architecture that relies on a *test system coordinator* (TSC) and several TT4RT instances. As outlined in Fig. 24.7, TSC is mainly charged with distributing selected test cases to rerun and assigning their corresponding test components to the execution nodes. Several TT4RT instances are installed within the host computers involved in the final execution environment. They can be seen as test containers that hold test components (i.e., either MTC or PTC components). Each instance controls the execution of a subset of selected test cases.

24.3 Runtime Testing of Behavioral Adaptations

Running old test suites on dynamic software systems, in which not only the structure evolves but also the behavior may change, seems to be meaningless. Therefore, it is highly required to update test suites in a cost-effective manner as long as the software system is changing to fulfill new requirements. In this section, we address this issue by merging model-based testing and selective regression testing capabilities. To do so, we propose a selective test generation approach, called TestGenApp.

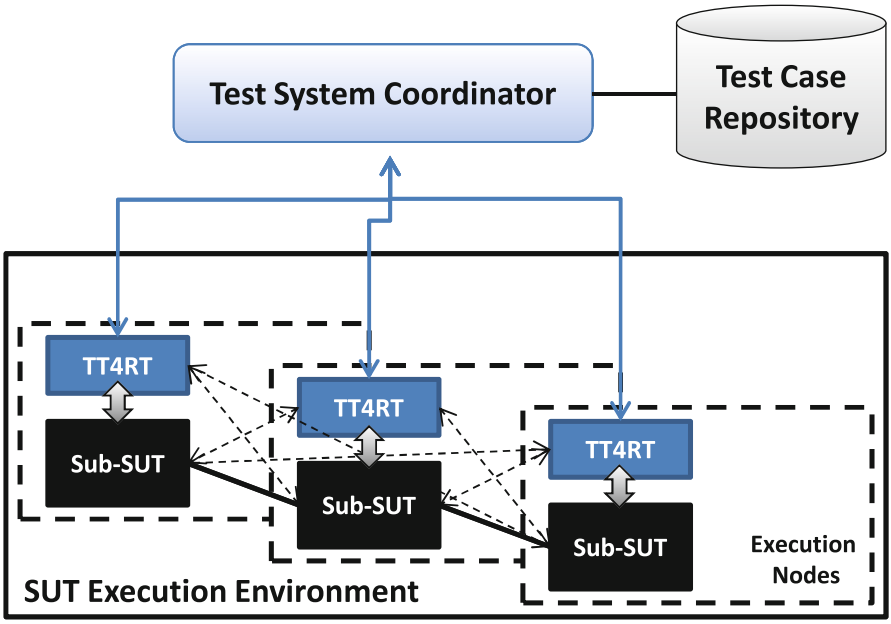


Fig. 24.7 The distributed test execution platform

As illustrated in Fig. 24.8, our selective test generation approach is composed of four modules.

- **Model differencing module:** It is proposed to capture correspondences and differences between two models in terms of added, removed, or modified locations and transitions.
- **Old test suite classification module:** It is charged with classifying the old test suite issued from the original model $\mathcal{M}$ into *reusable*, *retestable*, *aborted*, and *obsolete* tests.
- **Test generation and recomputation module:** It generates new abstract test sequences covering newly added behaviors and adapts aborted and obsolete tests.
- **TTCN-3 transformation module:** It is used to transform the abstract test sequences, obtained in the last step, into the TTCN-3 code.

24.3.1 Prerequisites: UPPAAL Timed Automata

In order to specify the behavioral models of evolved systems, *timed automata* (TA) is chosen for the reason that it is a widespread formalism usually used for modeling behaviors of critical and real-time systems. More precisely, we opt for the particular UPPAAL style [2] of timed automata because UPPAAL is a well-

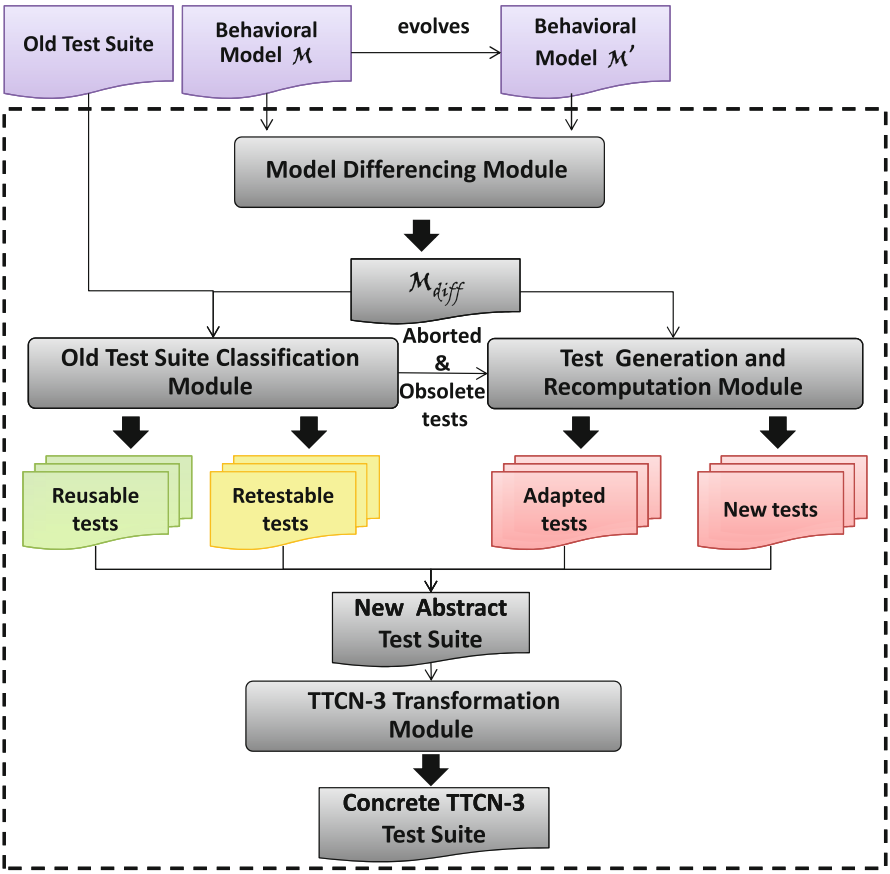


Fig. 24.8 TestGenApp: selective test case generation approach

established verification tool. It is made up of a system editor that allows users to edit easily timed automata, a simulator that visualizes the possible dynamic execution of a given system and a verifier that is charged with verifying a given model w.r.t. a formally expressed requirement specification. Within UPPAAL timed automata, a system is modeled as a network of timed automata, called processes. A timed automaton is an extended finite-state machine equipped with a set of clock-variables that track the progress of time and that can guard when transitions are allowed.

Let $\mathcal{C}$ be a set of variables called clocks, and $\mathcal{A}ct = \mathcal{I} \cup \mathcal{O} \cup \{\tau\}$ with $\mathcal{I}$ a set of input actions, $\mathcal{O}$ a set of output actions, and the non-synchronizing action (denoted τ). Let $\mathcal{G}(\mathcal{C})$ denote the set of guards on clocks being conjunctions of constraints of the form $c \bowtie n$, where $c \in \mathcal{C}$, $n \in \mathsf{N}$, and $\bowtie \in \{\leqslant, \leq, =, \geq, \geqslant\}$. Moreover, let $\mathcal{U}(\mathcal{C})$ denote the set of updates of clocks corresponding to sequences of statements of the form $c := n$.

A timed automaton over $(\mathcal{A}\text{ct}, \mathcal{C})$ is a tuple $(L, l_0, \mathcal{A}\text{ct}, \mathcal{C}, I, E)$, where:

- L is a set of locations, $l_0 \in L$ is an initial location.
- $I : L \longmapsto \mathcal{G}(\mathcal{C})$ a function that assigns to each location an invariant.
- E is a set of edges such that $E \subseteq L \times \mathcal{G}(\mathcal{C}) \times \mathcal{A}\text{ct}_\tau \times \mathcal{U}(\mathcal{C}) \times L$

We write $l \xrightarrow{g,\alpha,u} l'$ when $\langle l, g, \alpha, u, l' \rangle \in E$.

Let $(L, l_0, \mathcal{A}\text{ct}, \mathcal{C}, I, E)$ be a timed automaton. The semantics of TA is defined in terms of a timed transition system over states in the form (l, σ) where l is a location and $\sigma \in \mathbb{R}^{\mathcal{C}}_{\geqslant 0}$ is a clock valuation satisfying the invariant of l. The initial state (l_0, σ_0) is a state where l_0 is the initial location of the automaton and σ_0 is the initial mapping where $\forall c \in \mathcal{C}, c = 0$. Indeed, there are two kinds of transitions:

- Delay transitions, $(l, \sigma) \xrightarrow{d} (l, \sigma + d)$, in which all clock values of the automaton are incremented with the amount of the delay, denoted $\sigma + d$. In such a case, the automaton may stay in a location l as long as its invariant remains true.
- Discrete transitions, $(l, \sigma) \xrightarrow{\alpha} (l', \sigma')$, correspond to the execution of edges (l, g, α, u, l') for which the guard g is satisfied by σ. The clock valuation σ' of the target state is obtained by modifying σ according to updates u.

A run of timed automaton $(L, l_0, \mathcal{A}\text{ct}, \mathcal{C}, I, E)$ is a sequence of transitions $(l_0, \sigma_0) \xrightarrow{d_1} \xrightarrow{\alpha_1} (l_1, \sigma_1) \xrightarrow{d_2} \xrightarrow{\alpha_2} \cdots \xrightarrow{d_n} \xrightarrow{\alpha_n} (l_n, \sigma_n)$, with $\sigma_i \in \mathbb{R}^{\mathcal{C}}_{\geqslant 0}$, $d_i \in \mathbb{R}_{\geqslant 0}$ and $\alpha_i \in \mathcal{A}\text{ct}$. A network of timed automata, $TA_1 \| \ldots \| TA_n$ over $(\mathcal{A}\text{ct}, \mathcal{C})$ is modeled as a timed transition system obtained by the parallel composition of n TA over $(\mathcal{A}\text{ct}, \mathcal{C})$. Synchronous communication between the timed automata is performed by hand-shake synchronization using input and output actions.

24.3.2 Differencing Between Behavioral Models

We introduce a novel *differencing algorithm* that concisely captures differences and similarities between networks of timed automata. In such a case, two main elements are compared: locations and transitions. First, we differentiate automata at the transition level. The two transitions T_i in the initial $\mathcal{TA}$ and T_j in the evolved $\mathcal{TA}'$ are considered similar if the following conditions are met:

(a) T_i and T_j have the same source and target locations, and
(b) they have the same values in the guard, assignment, and synchronization fields.

The procedure used for this purpose takes as input two array lists including transitions of two timed automata: $\mathcal{TA}$ and $\mathcal{TA}'$. For each transition in the initial automaton, we firstly check its presence within the evolved one. From a technical point of view, this condition is checked by looking for an equivalent transition in the evolved model having similar source location id and target location id. As long as this condition is satisfied, we look for meeting conditions defined above meaning that they have the same source and target locations (i.e., name, label,

committed, and urgent) and unchanged transition labels (i.e., guard, assignment, and synchronization). As a result, the transition is considered unmodified.

If at least one condition is not respected, the transition is considered modified and it is marked in yellow (see lines 8–9). New transitions which exist only in the evolved model are finally marked in red (see lines 13–16). If a transition in $\mathcal{TA}$ does not have an equivalent in the new timed automaton $\mathcal{TA}'$, then this transition is not copied in the final array list because it is considered as a removed transition. The output of this procedure is an array list containing all marked transitions (unmodified, modified, and new ones).

Following the same logic, we compare locations in both models. Two locations l_i in $\mathcal{TA}$ and l_j in $\mathcal{TA}'$ are considered similar if the following conditions are satisfied:

(a) l_i and l_j have the same name and the same identifier,
(b) they have the same incoming and outgoing transitions, and
(c) they have the same invariant expression.

One location is marked as changed if at least one of these conditions is not met. Finally since the SUT is generally modeled by a network of timed automata, it is necessary to apply these procedures for each timed automaton in the network.

24.3.3 Old Test Suite Classification

Inspired from the test classification proposed by Leung et al. [14], we introduce in this section a new test classification algorithm in which the old test suite generated from the original model $\mathcal{M}$ is analyzed and then partitioned into:

- Reusable test set T_{Ru}: valid traces that traverse unimpacted items by the change.
- Retestable test set T_{Rt}: valid traces that traverse impacted items by the change.
- Aborted test set T_{Ab}: invalid traces that cannot be animated on the new model because they cannot traverse modified items.
- Obsolete test set T_{Ob}: invalid traces that cannot be animated on the new model because they traverse removed items.

For that aim, each trace in the $\mathcal{TR}$ set should be animated on the $\mathcal{M}_{diff}$ model and its covered items should be identified. Two scenarios are then tackled. On the one hand, the test animation on the new model is achieved successfully. If the $trace$ traverses unchanged items, it is classified as a *reusable test*. Otherwise, it is classified as a *retestable test*. On the other hand, the test animation on the new model is abandoned. If this abort is due to some removed items which are no longer available in the new model, the $trace$ is seen as an obsolete test and it should be automatically discarded from the new test suite. Otherwise, this abort can be due to a modified transition which cannot be reached any more. In such a case, the $trace$ is classified as an *aborted test*.

24.3.4 Test Generation And Recomputation

Our approach identifies critical regions in the evolved model not only by marking added locations and transitions but also by detecting old traces that cannot be animated on the new model. Consequently, the $\mathcal{M}_{diff}$ is used in this stage to generate new tests and adapt aborted and obsolete ones in a cost-effective manner.

To generate new tests covering newly added behaviors, we are based on the findings of Blom et al. [3], which express coverage criteria by using observer automata with parameters and formulate the test generation problem as a search exploration problem. Instead of adding auxiliary variables to enable the expression of a coverage criterion as a reachability property using UPPAAL, the superposition of an observer onto timed automata is supported.

The test generation tool UPPAAL CO$\surd$ER [7] supports the concept of observers and the test case generation algorithm [6]. This efficient test suite generator is adopted in this thesis to realize a selective test generation approach when behavioral adaptations occur. The key idea is to formulate an observer that monitors only new regions in the evolved model. A test sequence satisfies this coverage criterion if when executed on the model it traverses at least one new edge where the *col* variable is updated to zero.

24.3.5 Test Case Concretization

At this stage, we define several rules to derive TTCN-3 test cases from abstract test sequences (see Table 24.1) [20]. First of all, we assume that for each test suite, a TTCN-3 module should be generated (**R1**).

Within the TTCN-3 standard, the module concept is used as a top-level structure. The first part includes definitions of test data, templates, test components, functions, communication ports, test cases, and so on. The second part is usually used to describe the execution sequence of test cases. A test component can be either a main test component (MTC) or a parallel test component (PTC). Remember that the MTC is charged with creating PTC components and executing TTCN-3 test cases. To do so, a port must be defined in order to specify a *point of control and*

Table 24.1 TTCN-3 transformation rules

Rules	Abstract concepts	TTCN-3 concepts
R1	A test suite	A TTCN-3 module
R2	A single trace	A TTCN-3 test case
R3	Time dependent behavior	A timer definition
R4	A test sequence in the form of input delay output	A TTCN-3 test behavior
R5	Each involved TA	A PTC component
R6	Each channel	A template

observation via which the test component can interact with other components and with the SUT. To specify time delays, TTCN-3 supports a timer mechanism (**R3**). Timers can be declared in component type definitions, the module control part, test cases, functions, and altsteps. The channels declared in the UPPAAL XML file are transformed into TTCN-3 templates (**R6**).

Moreover, an abstract test system interface is defined similarly to a component definition. It includes a list of all possible communication ports through which the test system is connected to the SUT. Once the test configuration is generated, we look for the mapping of the abstract test sequences to test cases. As stated in Table 24.1, for each test behavior in the form of *input delay output* a TTCN-3 function is derived (**R4**). Moreover, for a single trace (i.e., an abstract test sequence), a test case is generated (**R2**). Then, the communication is established between the PTC ports and the system ports. Finally, a sequence of calls to the already generated TTCN-3 functions is performed. To compile the obtained test cases, the TThree compiler [19] is used. It transforms the *abstract test suite* into an *executable test suite*. Then, our TT4RT test system can be used for test isolation and execution purposes.

24.4 Conclusion

In this article, we applied the runtime testing process to validate dynamically adaptable IoT networks in smart cities after the occurrence of dynamic structural adaptations. For this aim, we proposed a generic and resource aware test execution platform that covers essentially two phases. The first phase deals with test selection and distribution concerns. The main issue tackled in this first part is alleviating test burden, cost, and resource consumption. This goal is achieved by reducing the amount of test cases to rerun and by assigning efficiently their associated test components to execution nodes while fitting resource and connectivity constraints. The second phase handles test isolation and execution concerns. Based on the TTCN-3 standard, we proposed a test system, TT4RT, which performs tests written in a standardized notation. Accordingly, we gained in terms of using the same notation for all types of tests and using a generic and flexible test harness. Furthermore, TT4RT afforded a test isolation infrastructure supporting components with various testability options (i.e., testable, test aware, untestable, etc.).

As another contribution to this article, we defined a model differencing technique that highlights similarities and differences between an original behavioral model and the evolved one, generally obtained after behavioral adaptations. Second, we provided a test classification technique that selects efficiently reusable and retestable tests, identifies aborted tests, and discards obsolete ones. These two steps are responsible for identifying critical regions in the evolved model that need to be covered by newly generated tests. For this purpose, we specified our own coverage criteria based on the observer automata language and we used the well-established tool UPPAAL model-checker and its extension UPPAAL CO$\surd$ER for generating

new tests. Also, a test recomputation technique was introduced with the aim of adapting aborted and obsolete tests. Finally, the mapping of the abstract test sequences to the TTCN-3 notation was handled.

Many possible extensions for our work are possible:

- **Meta-heuristic techniques for the constrained test placement problem:** The major problem we may face while applying our approach on large-scale environments comes from the constrained test placement module. In fact, this module requires a long time to compute an exact optimal solution fitting the resource and connectivity constraints. Therefore, we intend to use the *Tabu search* (TS) meta-heuristic as a resolution algorithm and performing a parallel exploration of the solution domain.
- **Extension of the distributed TTCN-3 test system:** The so far proposed approach focuses only on distributing TTCN-3 test cases. Each one is managed by a *main test component* (MTC) and may create several parallel test components (PTC) in order to execute integration tests. To gain more performance and to alleviate the test workload on the execution environment, we should also distribute PTC components over the execution nodes in order to avoid the communication overhead introduced by the centralized execution architecture [18].
- **Test generation based on probabilistic model-checking:** The key idea here is to apply runtime testing before the occurrence of dynamic proactive adaptations which consist in making predictions of how the environment or the system is going to evolve in the near future. To do so, tests have to be generated from behavioral models that are augmented with probabilities to describe the unpredictable system's behavior. Formalisms like *probabilistic timed automata* can be used to specify the system behavior.

References

1. Alhazbi, S., Jantan, A.: Dependencies management in dynamically updateable component-based systems. J. Comput. Sci. **3**(7), 499–505 (2007)
2. Behrmann, G., David, A., Larsen, K.G.: A tutorial on Uppaal. In: Bernardo, M., Corradini, F. (eds.) International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004. Revised Lectures. LNCS, vol. 3185, pp. 200–237. Springer, Berlin (2004)
3. Blom, J., Hessel, A., Jonsson, B., Pettersson, P.: Specifying and generating test cases using observer automata. In: Proceeding of the 5th International Workshop on Formal Approaches to Software Testing (FATES'05), pp. 125–139 (2005)
4. Chen, I.-Y., Tsai, C.-H.: Pervasive digital monitoring and transmission of pre-care patient biostatics with an OSGi, MOM and SOA based remote health care system. In: Proceeding of the 6th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'06), pp. 704–709 (2008)
5. Ghédira, K., Dubuisson, B.: Foundations of CSP. In: Constraint satisfaction problems, pp. 1–28. Wiley, New York (2013)
6. Hessel, A.: Model-Based Test Case Generation for Real-Time Systems. Ph.D. thesis, Uppsala University, Sweden (2007)

7. Hessel, A., Pettersson, P.: CO$\surd$ER a real-time test case generation tool. In: Proceeding of the 7th International Workshop on Formal Approaches to Testing of Software (FATES'07) (2007)
8. Ioannidis, Y.E., Rantakrishnan, R.: Efficient transitive closure algorithms. In: Proceedings of the 14th International Conference on Very Large Databases (VLDB'88) (1988)
9. Joachim, W.: Internet-of-Things architecture IoTA project deliverable D1.2 – Initial Architectural Reference Model for IoT 07 (2018)
10. Kienzle, J., Guelfi, N., Mustafiz, S.: Crisis management systems: a case study for aspect-oriented modeling. In: Transactions on Aspect-Oriented Software Development VII: A Common Case Study for Aspect-Oriented Modeling, pp. 1–22. Springer, Berlin (2010)
11. Lahami, M., Fakhfakh, F., Krichen, M., Jmaïel, M.: Towards a TTCN-3 test system for runtime testing of adaptable and distributed systems. In: Proceedings of the 24th IFIP WG 6.1 International Conference Testing Software and Systems (ICTSS'12), pp. 71–86 (2012)
12. Lahami, M., Krichen, M., Bouchakwa, M., Jmaïel, M.: Using Knapsack problem model to design a resource aware test architecture for adaptable and distributed systems. In: Proceedings of the 24th IFIP WG 6.1 International Conference Testing Software and Systems (ICTSS'12), pp. 103–118 (2012)
13. Larsson, M., Crnkovic, I.: Configuration management for component-based systems. In: Proceeding of the 10th International Workshop on Software configuration Management (SCM'01) (2001)
14. Leung, H.K.N., White, L.: Insights into regression testing [software testing]. In: Proceedings of the International Conference on Software Maintenance (ICSM'89), pp. 60–69 (1989)
15. Li, B., Zhou, Y., Wang, Y., Mo, J.: Matrix-based component dependence representation and its applications in software quality assurance. ACM SIGPLAN Not. **40**(11), 29–36 (2005)
16. Qu, B., Liu, Q., Lu, Y.: A framework for dynamic analysis dependency in component-based system. In: 2nd International Conference on Computer Engineering and Technology (ICCET'10), pp. 250–254 (2010)
17. Rothermel, G., Harrold, M.J.: Analyzing regression test selection techniques. IEEE Trans. Softw. Eng. **22**(8), 529–551 (1996)
18. Schieferdecker, I., Vassiliou-Gioles, T.: Realizing distributed TTCN-3 test systems with TCI. In: Proceedings of the 15th IFIP International Conference on Testing of Communicating Systems (TestCom'03) (2003)
19. Testing Technologies. TTthree - Compile TTCN-3 modules into test executables. http://www.testingtech.com/products/ (2008)
20. Vassiliou, T., Rennoch, A., Desroches, C., Schieferdecker, I.: TTCN-3 Quick Reference Card (2016)

Chapter 25
HCDSR: A Hierarchical Clustered Fault Tolerant Routing Technique for IoT-Based Smart Societies

Thaha Muhammed, Rashid Mehmood, Aiiad Albeshri, and Ahmed Alzahrani

25.1 Introduction

Urban population has increased vastly in the recent years. The United Nations Human Settlements Program (UN-Habitat) [41] has foreseen it to be 10 billion by 2050, which is two-thirds of the current population on earth. The cities will have to deal with pressing issues such as public safety, efficient transportation, energy consumption, environmental sustainability, and expense reduction. These pressing issues have led to smart city paradigm which aims to plan and develop efficient urban cities in future.

The past decade has witnessed the advancement of Internet of Things (IoT) [37, 40] especially the sensing technology [25]. In addition to the widespread development of sensors, improvement in big data computing infrastructure has enabled the collection of huge amount of heterogeneous data produced daily by urban spaces [11]. Urban spaces produce data related to temperature, weather, pollution, traffic control, the mobility of people, and resource consumption (water and electricity) which can be analyzed to improve the services provided and make the environment greener. Smart cities rely on sensors, webcams, IoT systems, wireless sensor networks, databases, ubiquitous devices, and many other frameworks that collect, process, and take informed decisions based on the data [5]. A survey on data fusion and IoT for smart ubiquitous environments can be seen in [6].

T. Muhammed (✉) · A. Albeshri · A. Alzahrani
Department of Computer Science, FCIT, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: m.thaha.h@ieee.org; aalbeshri@kau.edu.sa; asalzahrani@kau.edu.sa

R. Mehmood
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: RMehmood@kau.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_25

Wireless sensor networks (WSNs) are one of the atomic components of IoT. Data acquisition for IoT applications requires wireless sensor network and is the link between the real world and the digital world. WSNs play a major role in building interconnected urban territories and are critical to smart cities. WSNs consist of small low power sensor nodes that can sense, process, and wirelessly communicate with each other. The sensors are devices with limited battery, storage, size, and computational power. The sensors nodes sense data and forward it to a base station known as sink for further processing of data by IoT systems. Intelligent monitoring and management of smart cities are possible through IoT. WSNs are used in a number of time-critical smart city applications such as agriculture monitoring [43], intruder detection [19], disaster management, health care, mobile object tracking, environment monitoring [19], intelligent transport system (obstacle detection, collision warnings and avoidance, traffic monitoring) [16, 42, 47], vehicular ad-hoc networks [35], energy monitoring in smart grids [38], and home/office automation systems (HOS) [20].

Since WSNs are deployed in harsh and hostile conditions they are susceptible to frequent errors. The occurrence of faults results in disruption of the network or worse in the failure of the network. This might lead to human, economic, environmental loss since the sensors are used in many safety critical applications. Another source of a fault in WSN is the power [25]. Since the WSNs work unattended in a hostile environment it is not feasible to replenish the batteries of the sensors. Moreover, various hazard might cause the power to run out, which results in a node failure. Data transmission consumes a major portion of energy [46]. Hence prolonging energy in WSN becomes a critical and challenging issue [1, 3, 9, 15, 32]. A detailed discussion on possible faults in wireless sensor networks has been discussed in [39].

It is required that the data collected by the sensors on critical events should not be of low quality [18, 21, 22] that might lead to important information loss, but often random link failures occur that disrupt communication in the network. All these issues point to the necessity of fault tolerance techniques that would provide techniques to mask these faults and provide the expected services, in the presence of faults. Major disadvantages of existing techniques are a high dissipation of energy, large mean time to repair (MTTR), and the use of extra software and hardware [44, 45].

Clustering has been used by the researchers to reduce the energy consumption in WSN [4]. In a clustered WSN, the sensors are clustered into mutually exclusive clusters. A sensor node is associated with a single cluster and each cluster has a cluster head (CH) that aggregates the data from the nodes associated with it and transmits it to the base station (sink). The transmission of a large amount of data by the CHs leads to depletion of energy and consequently leads to the death of the CH. To prevent this many researchers [28–30] have proposed using special gateways with higher initial energy, but the use of special gateways is not feasible when the network is deployed randomly in inaccessible locations. This also creates a problem with clustering and reclustering when randomly deployed. Consequently, many researchers have used techniques such as multipath routing, backbone scheduling,

and node scheduling. Researchers have also proposed various clustering techniques to improve ad-hoc network performance, see, e.g., [7, 8, 33, 34, 36].

In this paper, we propose a new fault tolerant routing algorithm based on modified dynamic source routing (DSR) on a clustered, hierarchical sensor network for IoT applications. We use a vice cluster head that takes over the duties of the CH on the failure of a CH. Moreover, we use multiple paths that have been prioritized and sorted on the basis of a cost function that takes into consideration the total energy in a path and the distance from the source to sink. Furthermore, we use energy thresholds to decide the CHs that would participate in the routing process. One of the major advantages of the technique is that the mean time to repair (MTTR) for this technique is small. We simulate our algorithm and compare our algorithm with DFTR [12], a distributed fault tolerant algorithm and LEACH (low-energy adaptive clustering hierarchy) [25], a well-known routing algorithm. Metrics such as the number of alive nodes, total energy consumption of the network, and total packets transmitted to the sink are compared measured for all the three techniques. Based on these metrics it was observed that HCDSR performs better than the other techniques. This paper extends our earlier work [40]. The contributions to this work can be summarized as follows:

- We present a survey of fault tolerant and energy-efficient routing techniques for WSNs.
- We propose a new up-to-date taxonomy for fault tolerant strategies for WSNs.
- We provide a brief qualitative analysis and comparison of latest fault tolerant strategies for WSNs.
- We propose a new energy-efficient fault tolerant routing strategy called heterogeneous modified dynamic source routing (HCDSR).
- We simulate the proposed technique.
- The results from the proposed technique are compared with two current techniques, LEACH [25] and DFTR [12], demonstrating better performance.

The rest of the article is organized as follows. Section 25.2 discusses the proposed taxonomy for fault tolerant techniques in WSN. In Sect. 25.3, we discuss the state-of-the-art fault tolerant techniques for WSN. We also do a qualitative analysis of FT techniques in WSN. In Sect. 25.4, we discuss the system model and Sect. 25.5 introduces our proposed FT routing technique. Section 25.6 presents the simulation of the proposed technique. It also presents the comparison with techniques to validate our proposed technique. Section 25.7 concludes the paper.

25.2 Taxonomy

Fault tolerance techniques in wireless sensor networks can be classified according to two criteria, namely based on the phase at which the fault tolerant technique triggers and based on the origin of faults in WSN. Based on these criteria fault detection

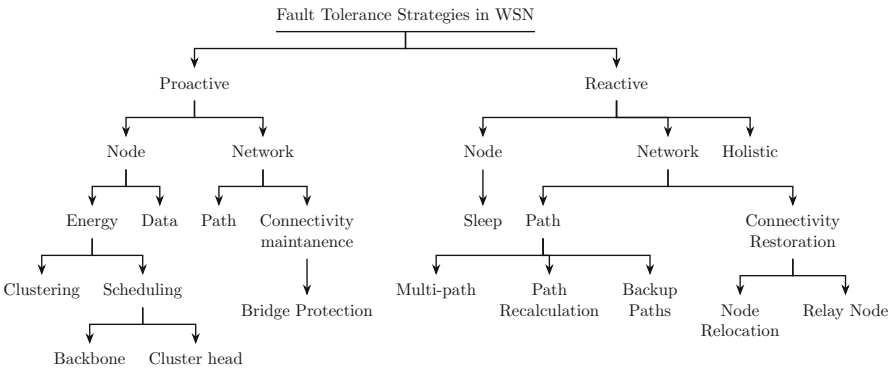


Fig. 25.1 Proposed taxonomy of fault tolerant techniques in WSN

techniques in WSN can be classified as (1) proactive and (2) reactive as shown in Fig. 25.1.

25.2.1 Proactive Techniques

Proactive techniques in WSN proactively and sensibly use the existing resources of the wireless sensor to extend the lifetime of the network or prevent the fault from occurring. These techniques take preemptory action against potential faults. Based on the origin of faults these techniques can be classified (1) node-based techniques, (2) network-based techniques, and (3) holistic techniques.

Node-Based Techniques The node-based proactive techniques can be further classified as (1) energy-based fault tolerance technique and (2) data fault tolerance. Energy-based fault tolerance increases the mean time to failure and the lifetime of the network. This strategy uses techniques such as clustering of sensor nodes, hibernation of nodes, and scheduling nodes and backbone of the WSN. Proactive data fault tolerant techniques help in recovering from data faults. One of the major techniques of data fault tolerance is the dual transmission of the same value and comparison of these data to detect the faults.

Network-Based Techniques It comprises of mainly two techniques, namely (1) connectivity maintenance technique and (2) multipath routing. Connectivity maintenance techniques increase the lifetime of network using various algorithms. Bridge protection algorithm is an example of connectivity maintenance algorithm that increases the lifetime of WSNs comprising bridged nodes. Data is sent through multiple paths to increase the redundancy and tolerate network fault in multipath techniques.

25.2.2 Reactive Techniques

Reactive techniques trigger the fault tolerant strategy on the occurrence of the faults. This strategy waits for the faults to occur and then adjusts or reacts to the fault by starting the recovery process. These techniques can also be further classified based on the origin of the faults as (1) node-based, (2) network-based, and (3) holistic-based technique.

Node-Based Techniques Node-based reactive techniques are used to recover from node failures. It consists of strategies like switching to the sleeping backup node on the occurrence of node failure.

Network-Based Faults Network-based reactive techniques consist of using multiple paths, backup paths, and path recalculation in case of network/link failure. Moreover, for restoring the connectivity, extra nodes are deployed or the existing nodes are repositioned.

Holistic Techniques These are the techniques that can deal and recover from both network- and node-based faults. They provide a complete fault tolerance for various faults.

25.3 Previous Work

In this section we shall discuss the existing work related to WSN fault tolerance techniques. In this section we shall discuss the current work related to WSN fault tolerance techniques. Tables 25.1 and 25.2, respectively, give the advantages and disadvantages of various fault tolerant techniques for WSNs that shall be discussed in this section. Tables 25.3 and 25.4 give a comparison of some of the FT routing techniques that shall be discussed ahead.

25.3.1 Proactive Techniques

Zhao et al. [49] propose a sleep scheduling technique, called virtual backbone scheduling (VBS). In this technique, we form multiple backbones that overlap with each other. Data is transmitted to the sink using only these backbones. The nodes that are not part of the backbone do not participate in transmission to save energy. The energy consumption of the nodes is balanced by rotating the backbones. This results in a longer lifetime of the network. Selection of the backbones that increases the network lifetime is an NP-hard problem and hence the authors propose three techniques to solve this. These schemes are based on (1) schedule transition graph (STG), (2) virtual scheduling graph (VSG), and (3) iterative local replacement (ILR). The longest path in a schedule transition graph corresponds to

Table 25.1 Comparison of advantages of various fault tolerant techniques in WSN

Protocol	Taxonomy	Advantages
VBS [49]	Proactive	– Energy-efficient routing – Increases lifetime of network
BPA [27]	Proactive	– Reduction in message overhead – Increases energy efficiency – Balance the energy in the network
PASC_AR [17]	Proactive	– Increases network lifetime
DFTR [12]	Proactive	– Reduces energy consumption – Increases gateway lifetime
MDSR [2]	Reactive	– Energy-efficient routing – Fault tolerant routing – Total throughput increases due to higher lifetime of the network
B^3FT [23]	Reactive	– Energy efficient – Energy balanced – Cluster head tolerance – Increases the time span of network (CH + nodes)
FT PSO [13]	Reactive	– Increases lifetime of network – Energy balanced
FTEAM [26]	Reactive	– Increases lifetime of cluster heads – Improves reliability
IFTF [24]	Holistic	– Holistic approach detects permanent node failures – Monitors network quality – Determines source and cause of the fault

Table 25.2 Comparison of disadvantages of various fault tolerant techniques in WSN

Protocol	Disadvantages
VBS [49]	– Node failures might require recalculating the backbones as nodes overlap among backbones
BPA [27]	– Trade-off between time and residual energy
PASC_AR [17]	– Reduction of sensing accuracy due to the sleep mode of nodes
DFTR [12]	– Fixed gateways not always suitable or plausible – Clustering difficult when there is fixed gateways
MDSR [2]	– Reduction in throughput of network for any given time period as all nodes not involved in transmission
B^3FT [23]	– Use of extra hardware as gateways
FT PSO [13]	– Does not handle fault tolerance if no gateway in range
FTEAM [26]	– Trade-off between accuracy and lifetime – Only reliable when rate of change of sensed value is very small inside the cluster
IFTF [24]	– Four percent increase in message overhead – Does not consider the computation overhead – Does not provide a specific technique for recovering from fault

Table 25.3 Comparison of fault tolerant routing techniques

Protocol	Energy savings	Phases	Message reduction	FT technique
MDSR [2]	✓	1	X	Multiple routes stored
E3BFT [23]	✓	3	✓	Rejoin new cluster head
FT PSO [13]	✓	3	✓	Selection of alternate gateways
DFTR [12]	✓	3	✓	Selection of alternate gateways

Table 25.4 Comparison of clustering techniques used in selected fault tolerant routing schemes

Protocol	Clustering	Clustering type	Intra cluster routing	Clustering technique
E3BFT [23]	Yes	Proactive	Multi-hop	Using residual energy, routing overhead, and node-gateway distance
MDSR [2]	X	NA	NA	NA
FT PSO [13]	Yes	Proactive	Multi-hop	Using special gateways
DFTR [12]	Yes	Reactive	Multi-hop	Using special gateways

the backbone that provides maximum network lifetime. STG models structure and energy separately. In VSG, multiple virtual nodes are created from sensor nodes in such a way that the energy of corresponding sensor nodes is represented by their degrees. STG and VSG are both centralized techniques. ILR is a distributed technique that uses local information for switching backbone. The switching is done node by node iteratively and each node decides the next node to be switched by analyzing the energy of neighboring nodes.

Khan et al. [27] propose a fault tolerant algorithm for bridge protection in WSNs. The fragmentation of WSN due to the various events might result in the formation of a bridge node, which maintains the network connectivity. Bridge nodes are nodes whose removal results in communication failure between the fragments of WSN. The authors propose a bridge protection algorithm with manifold goals. Primarily to prevent the bridge node(s) from prematurely exhausting the energy and secondarily for preventing the formation of new bridge nodes from its neighboring nodes and to maintain the minimal functionality of the network with minimal interference. The authors modify the functionalities at the bridge node, gate nodes (non-bridge neighboring nodes of bridge node(s) on the side without sink), and fan-out nodes (non-bridge neighboring nodes of bridge node(s) on the side with sink). Most message reasoning is shifted from sink to gate nodes. Heartbeat messages will no longer be sent by the nodes, instead the gate nodes will process and detect if a node fails and send a message only on failure to save energy. Occlusion reasoning is used to avoid obsolete and redundant messages being sent to sink to save energy. To avoid creating load on fan-out nodes, the bridge node splits the traffic based on round-robin scheduling. The fan-out nodes further split the traffic to avoid the traffic coalescing at a single node to result in energy depletion. The routing alternatives chosen by the fan-out and bridge nodes are selected using Delaunay triangulation of

the nodes and further reducing it into relative neighboring graphs, which will further diminish power consumption, extending the life of bridge nodes.

Boucetta et al. [17] propose an energy-efficient fault tolerant scheduling algorithm, called power aware scheduling and clustering protocol with adaptive redeployment (PASC_AR). In this technique, all nodes are considered to have the initial same energy level and same capabilities. The network is partitioned into zones. All sensor nodes which sense identical values are considered to be in the same zone. The network is clustered geographically based on node location. Only one node from each node will be active and this node will be assigned as the cluster head. The cluster head creates a TDMA schedule that is used to select a new cluster head. The cluster head is selected in rotation from the nodes in the zone based on this schedule. The rest of the nodes are put to sleep. The nodes at the sink tend to fail earlier due to higher traffic. To recover from this, the authors propose a cascading movement of the redundant sensor nodes toward zones near the sink. PASC_AR maintains network connectivity and coverage by preventing routing holes.

Azharuddin et al. [12] propose an energy saving and FT routing technique, called DFTR that not only deals with energy utilization of cluster heads but also their fault tolerance. The routing is done based on the following criteria: (1) gateway to next hop gateway distance, (2) next-hop gateway to base station distance, and (3) energy remaining at the next-hop gateway. In the cluster setup phase, the clusters are formed and data is transmitted to the base station by the cluster heads. In this technique, the cluster heads are classified into three categories: (1) forward cluster head (ForwardCH), (2) backward cluster head (BackwardCH), and (3) orphan cluster head (OrphanCH). ForwardCH (G) is a set of neighboring cluster heads of the cluster head G closer to the base station. BackwardCH (G) is a set of neighboring cluster heads of the cluster head G which are farther from base station than G and ForwardCH in different routes. OrphanCHs are cluster heads that are not included in ForwardCH and BackwardCH. If the next-hop faulty cluster head is from ForwardCH, then a new next-hop gateway from ForwardCH is selected on the basis of minimum cost. If the ForwardCH for a gateway is empty, then the new next-hop gateway is selected from BackwardCH on the basis of minimum cost. Moreover, if the gateway doesn't have any ForwardCH and BackwardCH, then that CH is said to be OrphanCH and a gateway that has BackwardCH can help out the OrphanCH.

25.3.2 Reactive Techniques

Rana [2] proposes a modified dynamic source routing (DSR) algorithm offering energy-efficient, fault tolerant routing. The major features of this technique are (1) non-usage of nodes below certain energy threshold in the routing process, and (2) two routes cached between source and destination. Initially in the route discovery phase, the source node floods the network with route request packets (RREQ). It appends to this packet, its energy level at time t. Only the nodes with energy

level above a threshold energy participate in the flooding process. Each node that participates in this process appends its energy level to the RREQ packet. Multiple packets traveling through multiple paths may reach the destination D. Two best paths are selected based on the highest average energy of the paths. The destination waits for a time period T, for the arrival of all possible packets before making the final decision. Once the decision has been made the destination sends a route reply packet (RREP) informing the source about the selected paths. In the case of failure in the primary path, the secondary path is immediately used and the neighboring nodes send route error message (RERR) to update the routing information.

Gupta et al. [23] propose an energy-efficient fault tolerant clustering algorithm, named B^3FT. In this technique, the authors discuss fault tolerance for cluster heads without the redundant usage of cluster heads. Initially during the bootstrap process, all the cluster heads broadcast a HELLO message which consists of the cluster head ID, distance from base station, remaining energy, and distance from the base station. Sensor nodes that receive this message are considered to be in the range of the cluster head. All nodes that did not receive the HELLO packet broadcast an REQ message. Nodes which receive the REQ packet reply with RES packet that consists of its ID, distance from the base station, and the overhead of cluster head it is the associated with. In the next phase, the sensors join various cluster heads by considering the routing overhead, the distance between the nodes and the gateways, and the remaining energy of relay nodes. If a sensor node couldn't find a cluster head, then the neighbor node with the higher remaining energy and minimal overhead acts as a cluster head for that node. In the case of cluster head failure, all sensor nodes associated with this node broadcast REQ message and wait for an REP message. If a node receives REP message from cluster heads, then the node will associate itself with cluster head that has minimum cost else it joins the node with minimum cost.

Azharuddin et al. [13] propose a fault tolerant clustering- based routing algorithm based on particle swarm optimization. This routing technique has two stages: (1) network setup stage and (2) steady state. During the network setup phase, the base station assigns ID to all the nodes and gateways. The nodes broadcast these IDs and are assigned to gateways depending on the distance from the nodes. The gateways send this local node information to base station, where the load (number of packets received) of each gateway is calculated. This can be calculated as follows:

$$P_r(g_i) = \begin{cases} \sum\{P_r(g_j) = g_i,\ g_j \in G\} & \text{If } NextHop(g_j) = g_i,\ \forall g_{j \in G} \\ 0 & \text{Otherwise} \end{cases} \tag{25.1}$$

where $P_r(g_i)$ is the number of data packets received by each gateway g_i per round.

Let E_R, $E_T(g_i, NextHop(g_i))$, and $E_{intraclstr}$ be the energy utilization due to receiving of data by the gateways, sending of a data packets to the next-hop gateway, and energy utilization of a gateway g_i due to various intra-cluster activity. The power utilized by a gateway g_i for a round can be calculated as follows:

$$E_{consump}(g_i) = P_r(g_i) \times E_R + (P_r(g_i) + 1) \times E_T(g_i, NextHop(g_i)) + E_{intraclstr}$$

The lifetime of g_i with remaining residual energy $E_r(g_i)$ can be calculated as follows: $Lifetime(g_i) = E_r(gi)/E_{consmp}(g_i)$. We maximize the lifetime of the gateway with minimum lifetime by minimizing the routing load over the gateway. This is achieved with the help of particle swarm optimization. In case of a gateway failure during routing, the preceding gateway to the failed gateway broadcasts a HELP message. The neighboring gateways respond and the gateway with maximum lifetime toward the base station is selected as the new gateway. If no gateways respond, then the gateway is assumed as dead.

Hezaveh et al. [26] propose a technique called fault tolerant and energy aware mechanism (FTEAM). In this technique, we identify overlapped sensor nodes and put the nodes with highest residual energies to sleep so they can be used as a cluster head in case of cluster head failures. FTEAM consists of four phases: (1) cluster formation, (2) error free, (3) cluster failure, and (4) error recovery phase. In the initial phase, clusters are formed as nodes associate themselves with cluster heads depending upon the distance. The cluster heads determine the overlapped nodes with similarly sensed data and put the nodes with higher residual energy to sleep. In the error free state, the nodes send sensed data to cluster heads, which aggregates the data and sends it to the base station. During this state many nodes die and the energy of the cluster heads gradually decline below a certain threshold. This results in switching to error recovery phase wherein the sleeping nodes are awoken and the 5% of nodes with the highest energy is chosen as new cluster heads. When all sleeping nodes die later, the network fails.

25.3.3 Holistic Techniques

Dima et al. [24] propose an integrated fault tolerance framework (IFTF) which holistically considers all the fault issues in WSNs. IFTF monitors the wireless sensor network and can detect and diagnose application level faults, network layer faults, and establish the root cause of the fault. To achieve this IFTF uses two services, (1) a network level diagnosis sub-service for identifying network level faults such as node and link failure and (2) an application testing sub-service for detecting application anomalies. IFTF manager coordinates these two sub-services together to deal with complex fault scenarios. Application testing sub-service is responsible for ascertaining the adeptness of the system in accomplishing its functionalities. It performs functional testing by comparing an input value with expected value. The network diagnosis service monitors the energy levels of the nodes and the connectivity of the nodes to their neighbors. It uses a two-phase detection algorithm for detecting permanently faulty nodes.

25.4 Network and Radio Model

We consider a clustered WSN which consists of a single base station/sink and multiple clusters of sensor nodes. The sensor nodes in each cluster are normal nodes that are responsible for sensing and transmitting the data to their respective cluster heads (CH). All the nodes and the cluster heads are considered to be homogeneous with identical initial energy levels. The CHs are also normal nodes with the same energy constraints as that of sensing nodes. The CHs receive the sensed data, aggregate the data, and forward it to the base station. Direct data transmission occurs if the base station is one hop away from the CH else the aggregated data is forwarded to the CH closer to the base station. The nodes are deployed randomly as in smart dust model. The sensor nodes and CHs are considered immobile. There is only a single base station which is stationary and has an inexhaustible power supply. All sensor nodes have equivalent bi-directional communication range. All the wireless links are assumed to be symmetric so as to compute the distance between the nodes based on the received signal strength [48]. CSMA/CA MAC protocol is used by the CHs for communicating with base station [48]. For energy consumption analysis we only consider the energy used due to transmission and receiving of data since radio is the most power consuming parts as the consumption due to sensing and computing is negligible.

In this technique we use a radio model that is used in [25]. The energy dissipated E_T due to the transmission of a message of size l-bit between two nodes separated by a distance d is given by

$$E_T(l,d) = \begin{cases} l(E_{elec} + \varepsilon_{FS} \times d^2) & for\ (d < d_0) \\ l(E_{elec} + \varepsilon_{MP} \times d^4) & for\ (d > d_0) \end{cases} \tag{25.2}$$

where $d_0 = \sqrt{\varepsilon_{FS}/\varepsilon_{MP}}$, E_{elec} is the electronic energy required by the circuit, ε_{FS} and ε_{MP} are the transmit amplifier parameters that represent the energy required by the amplifier in free space and multipath models, respectively. The energy dissipation at the receiver sensor node for a message of size l-bit is given by

$$E_R(l) = l \times E_{elec} \tag{25.3}$$

Moreover, the energy consumed for fusing $l-$bits can be given by

$$E_F(l) = l \times E_{df} \tag{25.4}$$

where E_{df} is the energy incurred due to fusing of one bit data.

25.5 Proposed Technique

The proposed technique has four phases: (1) setup phase, (2) route determination phase, (3) data communication phase, and (4) fault recovery phase.

25.5.1 Network Setup

Initially the network will be in the setup phase and all the sensor nodes send a HELLO message to the sink. The sink then assigns an ID to all sensor nodes. During the setup phase, we use any of the standard clustering algorithm to cluster the network and assign a cluster head to each cluster. The cluster head in each of the cluster sends a HELLO message to all its nodes with specific power and based on the strength of the signal received, it finds the nearest node to itself. The nearest node to the cluster head in each cluster is assigned as the vice cluster head. The sink then broadcasts a HELLO message using specific amount of power to all the cluster heads. The sink calculates the distances to each sensor node using the radio strength and this distance is sent back to the cluster head. The setup phase ends and the communication phase starts wherein the nodes send their data to the cluster heads and the cluster heads will fuse multiple identical values into a single value [25]. After a certain amount of time the network switches back to the setup phase so as to balance the energy of the nodes in the network. Subsequently, it enters the communication phase and this process continues until the network encounters a fault.

25.5.2 Route Discovery and Routing Algorithm

We develop our routing technique on top of the foregoing medium access control (MAC) layer. The major steps in our routing technique are given below:

Step 1: Initially in the route determination phase, each cluster head broadcasts an REQ packet similar to that of dynamic source routing (DSR). The REQ initially consists of the source ID, destination ID, the energy of the each cluster head, and the distance to the sink that was obtained during the bootstrap process.

Step 2: This REQ packet is flooded among other cluster heads, and each cluster adds to the packet their respective ID, energy level, and the distance to the base station.

Step 3: We define an energy threshold level. Any cluster head that has energy level below this threshold will not participate in the flooding process.

Step 4: The broadcasted REQ packets reach the destination. For each cluster head, the sink starts a timer on the arrival of the first REQ packet from that cluster head. The sink will wait for more packets till timer expires. Once the timer expires

the sink will analyze and select the routes for each cluster head based on the remaining energy level and the sum of distance of all cluster heads in the path to the sink. Based on this the routes for each cluster head are prioritized and are given priority numbers P1, P2,. . ., Pn.

Step 5: Thereafter, the sink sends an REP message to the cluster heads through all the discovered routes for the cluster heads. The REP message consists of the ID of the nodes that are in the path and the priority of that path.

Step 6: Once all REP messages reach the cluster heads they save the routes on basis of their priority and the route with priority P1 will be used for sending data to base station. The intermediate nodes between the source and destination also save the routes.

Step 7: Once the routes are selected, the cluster heads pass the route information to their respective vice cluster heads and the vice cluster head resume their sleep state after storing this information.

The routing algorithm has been given in Algorithm 1.

25.5.3 Fault Tolerance

In this technique we only consider faults in routing especially disruption of route due to failure of cluster heads. We can consider the following cases of failures:

Failure of the source cluster head. When the source cluster head fails, the vice cluster head takes over the job of the cluster head. The routing table is already present in the vice cluster head as explained before.

Failure of the intermediate cluster head. When the data from the source cluster head is sent ahead and one of the intermediate cluster heads fails then the failed cluster head sends an error message (ERR) to the preceding cluster head. The preceding cluster head will switch its route from primary to secondary route and the faulty cluster head will be replaced by the vice cluster head. If the secondary route also fails, then it will use the tertiary route and so on. Since the routes have been stored this will save us from recalculating the routes again.

Failure of vice cluster head. On the instance of vice cluster head failure, we go back to the network setup phase, recluster the network, and determine new routes. The fault tolerance algorithm has been given in Algorithm 2.

25.6 Simulation Results and Discussion

25.6.1 Experimental Setup

The proposed protocol was simulated using MATLAB R2015a on an Intel i5 machine with 2.40 GHz and 16 GB RAM running on Ubuntu 15.10. We deployed

Algorithm 1 Proposed routing algorithm

Input: $\forall CH_k,\ Energy_k,\ Distance\ to\ Sink_k$
Output: $All\ paths,\ from\ Source\ CH_i\ to\ Sink$

procedure CH–ROUTESELECTION
  $Node_i$ receives REQ_i packet from it $NeighborNode(i)$
  if $Node_i != sink$ **then**
    if $Energy(i) < E_thresh$ **then**
      $REQ_i \leftarrow REQ_i + (Id_i, Energy_i, Dist_i)$
      Forward REQ_i to $NeighborNodes(i)$
    else
      $Node_i$ does not broadcast
    end if
  else if $Node_i$ is == Sink **then**
    Start $timer_j$
    while $timer_i < Time_{thresh}$ **do**
      if $REQ == REQ_i$ **then**
        $REQSet(i) \leftarrow REQSet(i) \cup REQ_i$
      end if
    end while
    for each Request $REQ_i \in REQSet(j)$ **do**
      $Cost(j, i) \leftarrow 0.3 \times Dist(Source, Sink) + 0.7 \times Energy(Source, Sink)$
    end for
    for each row $Cost(j, :)$ **do**
      Sort $Cost(j, :)$
      Set Priority in Descending Order in $Cost(j, :)$
    end for
    for each REQ_i in $REQSet(j)$ **do**
      $REP_i \leftarrow REP_i + (Id_i, Path_i, Priority_i)$
      Forward REP_i to $Source_i$
    end for
    Node N_i creates routing table using REP messages.
  end if
end procedure

procedure CH–ROUTING
  Use the Path with $Priority = 1$
end procedure

400 sensor nodes in a square area of size 300×300 square meters. The topology of the simulated network is illustrated in Fig. 25.2. The sensor nodes were considered to have a starting energy of 2 J. When the energy level of the node reached 0 J the node was considered dead. We use Weibull reliability function [10] to model the faults in the cluster heads in our network which is given by

$$R(t) = e^{-\left(\frac{t-\gamma}{\eta}\right)^{\beta}} \tag{25.5}$$

where γ is the location parameter, η is the scale parameter, and β is the shape parameter. We set the values of $\gamma = 0$, $\beta = 3$, and $\eta = 3000$. If β is greater than 1, then the rate of failure increases with time else if β is less than 1, then the rate of failure decreases with time. Moreover, $\beta = 0$ the rate of failure is constant.

Algorithm 2 Fault tolerance algorithm

```
1: procedure CH–FaultTolerance
2:     for i ∈ Priority do
3:         if Path with Priority = i fails due to CH failure then
4:             if Vice CH not used then
5:                 Awake Vice CH
6:                 Replace CH with Vice CH
7:                 Update routing Tables
8:             end if
9:         else
10:            Use path with Priority = i + 1
11:        end if
12:    end for
13: end procedure
```

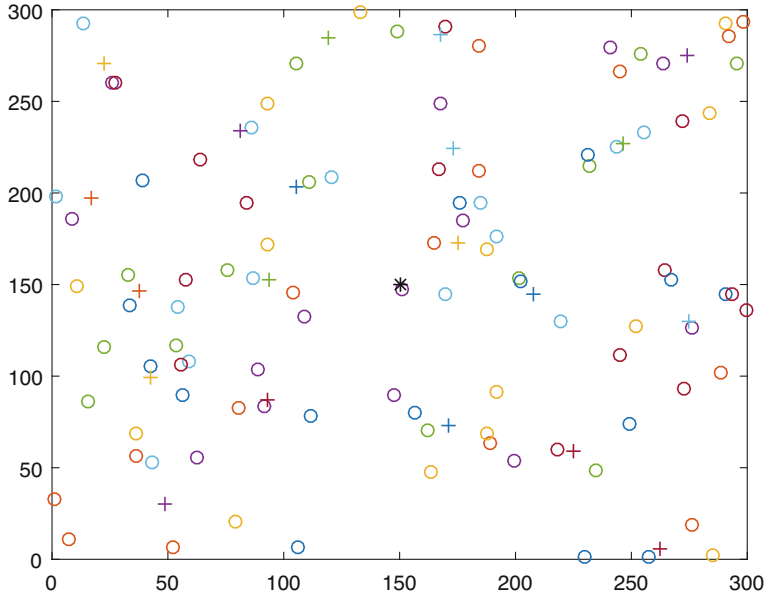


Fig. 25.2 The simulated network topology

β is chosen as per the analysis provided in [31] where it is established that the failure of cluster heads can be represented using a Weibull distribution with $\beta = 3$. Furthermore, [12] uses $\beta = 3$ for gateway faults in WSN. The simulation parameters used in the simulation are shown in Table 25.5. The parameters used are similar to [25].

The proposed algorithm is compared with DFTR [12] and LEACH [14] in terms of residual energy, number of packets received at sink, number of dead cluster heads, and network lifetime. We discuss the results of the experiments in the following sections.

Table 25.5 The simulation parameters

Simulation parameters	
Network size	400
Number of clusters	300×300
Initial sensor node energy	2.0 J
E_{elec}	50 nJ/bit
E_F	5 nJ/bit
Communication range	100 m
ε_{FS}	10 pJ/bit/m^2
ε_{MP}	0.0013 pJ/bit/m^4
d_0	88 m
Packet size	4000 bits
Message size	200 bits
E_{thresh}	20%

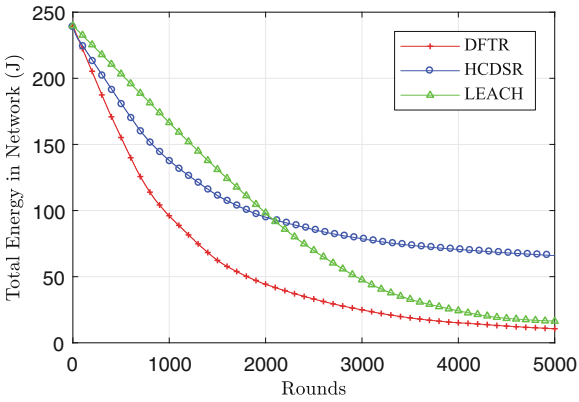


Fig. 25.3 The number of alive nodes per round in the simulated network among DFTR, LEACH, and HCDSR

25.6.2 Analysis of HCDSR

A wireless sensor network consisting of 120 nodes was simulated and clustered initially into 20 clusters. These 120 nodes were deployed in a sensing field of size 300 × 300. The sink was placed at the center of the sensing field at the coordinates (150,150). The simulated network is depicted in Fig. 25.2. The total number of alive nodes is compared in Fig. 25.3. We can see that the total alive nodes after 5000 rounds for the proposed technique are more than the LEACH and DFTR. Nodes are considered dead when their energy reaches 0 J or due to the simulation of faults following the Weibull distribution. We can observe in our proposed technique that initially there is a decrease in alive nodes that stabilize after a certain amount of rounds.

The stability of alive nodes is due to the energy threshold that was applied which resulted in many cluster heads with lower energy not to participate in the clustering, whereas in LEACH protocol, we can observe that the rate of dead nodes increases after a certain number of rounds. The DFTR protocol that does not provide an energy

threshold has the least amount of total energy in the network. This is because once the cluster head dies in DFTR technique and nodes of clusters become orphan they have to send it to a longer distance. Since DFTR uses special fixed gateways, a failure in the cluster head means another normal node doesn't take its place as a replacement, unlike the proposed technique. Figure 25.4 shows the total energy in the network per round for each of the technique. It is also similar to the previous graph where the total energy at the end of 5000 rounds is highest in the proposed technique. Hence, we can clearly say that the proposed technique increases the overall lifetime of the network as compared to LEACH and DFTR.

The total number of packets that have been transmitted to the sink for 5000 rounds has been compared in Fig. 25.5. We can see that the proposed technique transmits the maximum amount of packets. This is due to the presence of vice cluster heads which replace the failed cluster heads unlike LEACH protocol or DFTR protocol. A higher number of packets transmitted to the base station indicates the longer life of gateways. We can see that the proposed algorithm performs better than both DFTR and LEACH.

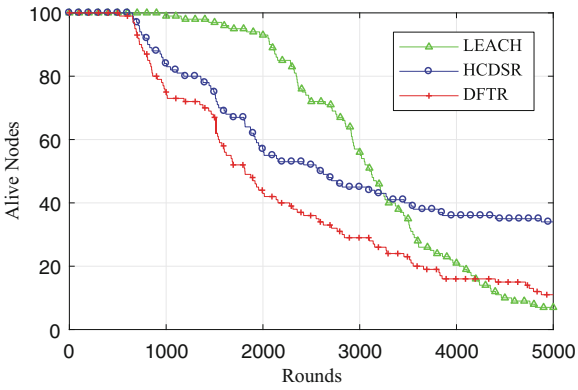


Fig. 25.4 Comparison of total energy in the simulated network per round among DFTR, LEACH, and HCDSR

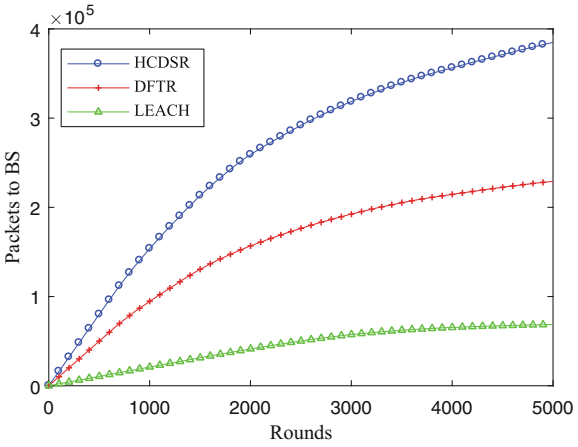


Fig. 25.5 The total number of data packets received at sink per round among DFTR, LEACH, and HCDSR

25.7 Conclusions

In this article, we have proposed a reliable and resilient routing technique for wireless sensor networks that forms the atomic component of IoT for smart city applications. We have proposed a taxonomy for fault tolerant techniques in WSN. Furthermore, we proposed a new fault tolerant routing algorithm for hierarchical WSN networks based on modified DSR (dynamic source routing) and vice cluster heads. Multiple routes are identified and these routes are prioritized on the basis of residual energy in the path and the distance of the source from the sink. In addition, the proposed technique uses vice cluster heads to tolerate faults during routing. We have shown through simulation that the proposed technique is better than LEACH and DFTR in terms of total energy in the network, the total number of packets transmitted to the sink, and the number of alive nodes. Our future work will be based on the mobility of the sensor nodes and the interaction with other IoT components.

Acknowledgements The authors acknowledge with thanks the technical and financial support from the Deanship of Scientific Research (DSR) at the King Abdulaziz University (KAU), Jeddah, Saudi Arabia, under the grant number G-651-611-38. The work carried out in this paper is supported by the High Performance Computing Center at the King Abdulaziz University, Jeddah.

References

1. Abbasi, A.A., Younis, M.: A survey on clustering algorithms for wireless sensor networks. Comput. Commun. **30**(14–15), 2826–2841 (2007)
2. Ahmed, R.E.: A fault-tolerant, energy-efficient routing protocol for wireless sensor networks. In: 2015 International Conference on Information and Communication Technology Research (ICTRC), pp. 175–178 (2015)
3. Akkaya, K., Younis, M.: A survey on routing protocols for wireless sensor networks. Ad Hoc Netw. **3**(3), 325–349 (2005)
4. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A survey on sensor networks. IEEE Commun. Mag. **40**(8), 102–114 (2002)
5. Alam, F., Mehmood, R., Katib, I., Albeshri, A.: Analysis of eight data mining algorithms for smarter internet of things (IOT). Procedia Comput. Sci. **98**(Suppl. C), 437–442 (2016). http://www.sciencedirect.com/science/article/pii/S187705091632213X
6. Alam, F., Mehmood, R., Katib, I., Albogami, N.N., Albeshri, A.: Data fusion and IoT for smart ubiquitous environments: a survey. IEEE Access **5**, 9533–9554 (2017)
7. AlTurki, R., Mehmood, R.: Multimedia ad hoc networks: performance analysis. In: 2008 Second UKSIM European Symposium on Computer Modeling and Simulation, pp. 561–566 (2008)
8. Alturki, R., Mehmood, R.: Cross-layer multimedia QoS provisioning over Ad Hoc networks. In: Using Cross-Layer Techniques for Communication Systems, pp. 460–499. IGI Global, Hershey (2012). http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-4666-0960-0.ch019
9. Anastasi, G., Conti, M., Di Francesco, M., Passarella, A.: Energy conservation in wireless sensor networks: a survey. Ad Hoc Netw. **7**(3), 537–568 (2009)
10. Antle, C.E., Bain, L.J.: Weibull distribution. In: Encyclopedia of Statistical Sciences, vol. 12, pp. 7629–7634. Wiley, Hoboken (2004)

11. Arfat, Y., Aqib, M., Mehmood, R., Albeshri, A., Katib, I., Albogami, N., Alzahrani, A.: Enabling smarter societies through mobile big data fogs and clouds. Procedia Comput. Sci. **109**, 1128–1133 (2017). http://www.sciencedirect.com/science/article/pii/S1877050917311213
12. Azharuddin, M., Jana, P.K.: A distributed algorithm for energy efficient and fault tolerant routing in wireless sensor networks. Wirel. Netw. **21**(1), 251–267 (2015)
13. Azharuddin, M., Jana, P.K.: A PSO Based Fault Tolerant Routing Algorithm for Wireless Sensor Networks, pp. 329–336. Springer, New Delhi (2015)
14. Balakrishnan, H., Heinzelman, W.R., Chandrakasan, A.: Energy-efficient communication protocol for wireless microsensor networks. In: 2014 47th Hawaii International Conference on System Sciences, vol. 08, 8020 (2000)
15. Bogliolo, A., Lattanzi, E., Acquaviva, A.: Energetic sustainability of environmentally powered wireless sensor networks. In: Proceedings of the 3rd ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor and Ubiquitous Networks, PE-WASUN '06, pp. 149–152. ACM, New York (2006)
16. Bottero, M., Chiara, B.D., Deflorio, F.: Wireless sensor networks for traffic monitoring in a logistic centre. Transp. Res. C Emerg. Technol. **26**, 99–124 (2013)
17. Boucetta, C., Idoudi, H., Saidane, L.A.: Adaptive scheduling with fault tolerance for wireless sensor networks. In: Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st, pp. 1–5. IEEE, Piscataway (2015)
18. Calero, C., Caro, A., Piattini, M.: An applicable data quality model for web portal data consumers. World Wide Web **11**(4), 465–484 (2008)
19. Casey, K., Lim, A., Dozier, G.: A sensor network architecture for tsunami detection and response. Int. J. Distrib. Sens. Netw. **4**(1), 27–42 (2008)
20. Cetinkaya, O., Akan, O.B.: Use of wireless sensor networks in smart homes. In: Emerging Communication Technologies Based on Wireless Sensor Networks: Current Research and Future Applications, pp. 233–258 (2016)
21. Gelenbe, E., Ngai, E.: Adaptive random re-routing for differentiated QOS in sensor networks. Comput. J. **53**(7), 1052–1061 (2010)
22. Gillies, D., Thornley, D., Bisdikian, C.: Probabilistic approaches to estimating the quality of information in military sensor networks. Comput. J. **53**(5), 493–502 (2010)
23. Gupta, S.K., Kuila, P., Jana, P.K.: E3bft: energy efficient and energy balanced fault tolerance clustering in wireless sensor networks. In: 2014 International Conference on Contemporary Computing and Informatics (IC3I), pp. 714–719 (2014)
24. Hamdan, D., Aktouf, O.E.K., Parissis, I., El Hassan, B., Hijazi, A.: Integrated fault tolerance framework for wireless sensor networks. In: 2012 19th International Conference on Telecommunications (ICT), pp. 1–6. IEEE, Piscataway (2012)
25. Heinzelman, W.B., Chandrakasan, A.P., Balakrishnan, H.: An application-specific protocol architecture for wireless microsensor networks. IEEE Trans. Wirel. Commun. **1**(4), 660–670 (2002)
26. Hezaveh, M., Shirmohammdi, Z., Rohbani, N., Miremadi, S.G.: A fault-tolerant and energy-aware mechanism for cluster-based routing algorithm of WSNs. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 659–664 (2015)
27. Khan, S.A., Bölöni, L., Turgut, D.: Bridge protection algorithms a technique for fault-tolerance in sensor networks. Ad Hoc Networks **24**, 186–199 (2015)
28. Kimençe, Ş., Bekmezci, İ.: Weighted relay node placement for wireless sensor network connectivity. Wirel. Netw **20**(4), 553–562 (2014)
29. Kuila, P., Jana, P.K.: Improved load balanced clustering algorithm for wireless sensor networks. In: Thilagam, P.S., Pais, A.R., Chandrasekaran, K., Balakrishnan, N. (eds.) Proceedings of the 2011 International Conference on Advanced Computing, Networking and Security, ADCONS'11, pp. 399–404. Springer, Berlin (2012)
30. Kuila, P., Jana, P.K.: Approximation schemes for load balanced clustering in wireless sensor networks. J. Supercomput. **68**(1), 87–105 (2014)

31. Lee, J.J., Krishnamachari, B., Kuo, C.C.J.: Aging analysis in large-scale wireless sensor networks. Ad Hoc Netw. **6**(7), 1117–1133 (2008)
32. Li, Y., Xiao, G., Singh, G., Gupta, R.: Algorithms for finding best locations of cluster heads for minimizing energy consumption in wireless sensor networks. Wirel. Netw. **19**(7), 1755–1768 (2013)
33. Mehmood, R., Alturki, R.: A scalable multimedia QoS architecture for ad hoc networks. Multimed. Tools Appl. **54**(3), 551–568 (2011). https://doi.org/10.1007/s11042-010-0569-0
34. Mehmood, R., Alturki, R.: Video QoS analysis over Wi-Fi networks. In: Advanced Video Communications over Wireless Networks, pp. 439–480. CRC Press, Boca Raton (2013)
35. Mehmood, R., Nekovee, M.: Vehicular ad hoc and grid networks: discussion, design and evaluation. In: Proceedings of the 14th World Congress on Intelligent Transport Systems (ITS), Beijing (2007)
36. Mehmood, R., Alturki, R., Faisal, M.: A scalable provisioning and routing scheme for multimedia QoS over ad hoc networks. In: Mauthe, A., Zeadally, S., Cerqueira, E., Curado, M. (eds.) Future Multimedia Networking, pp. 131–142. Springer, Berlin (2009)
37. Mehmood, R., Alam, F., Albogami, N.N., Katib, I., Albeshri, A., Altowaijri, S.M.: Utilearn: A personalised ubiquitous teaching and learning system for smart societies. IEEE Access **5**, 2615–2635 (2017)
38. Morello, R., Mukhopadhyay, S.C., Liu, Z., Slomovitz, D., Samantaray, S.R.: Advances on sensing technologies for smart cities and power grids: a review. IEEE Sens. J. PP(99), 1–1 (2017)
39. Muhammed, T., Shaikh, R.A.: An analysis of fault detection strategies in wireless sensor networks. J. Netw. Comput. Appl. **78**(Suppl. C), 267–287 (2017). http://www.sciencedirect.com/science/article/pii/S1084804516302545
40. Muhammed, T., Mehmood, R., Albeshri, A.: Enabling reliable and resilient IOT based smart city applications. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 169–184. Springer, Cham (2018)
41. Nations, U.: Global Urban Observatory (GUO) UN-Habitat, https://unhabitat.org/urban-knowledge/guo/
42. Nikam, S.S., Mane, P.B.: Swarm Intelligent WSN for Smart City, pp. 691–700. Springer, Singapore (2017)
43. Pantazis, N., Nikolidakis, S.A., Vergados, D.D.: Energy-efficient routing protocols in wireless sensor networks: a survey. IEEE Commun. Surv. Tutorials **15**(2), 551–591 (2013)
44. Ramanathan, N., Kohler, E., Girod, L., Estrin, D.: Sympathy: a debugging system for sensor networks [wireless networks]. In: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, pp. 554–555. IEEE Computer Society, Washington, DC (2004)
45. Ringwald, M., Römer, K., Vitaletti, A.: Snif: sensor network inspection framework. Tech. Rep. 535, Department of Computer Science, ETH Zurich, Zurich (2006)
46. Shnayder, V., Hempstead, M., Chen, B.R., Allen, G.W., Welsh, M.: Simulating the power consumption of large-scale sensor network applications. In: Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems, SenSys '04, pp. 188–200. ACM, New York (2004)
47. Wang, R., Zhang, L., Sun, R., Gong, J., Cui, L.: Easitia: a pervasive traffic information acquisition system based on wireless sensor networks. IEEE Trans. Intell. Transp. Syst. **12**(2), 615–621 (2011)
48. Xu, J., Liu, W., Lang, F., Zhang, Y., Wang, C.: Distance measurement model based on RSSI in WSN. Wirel. Sens. Netw. **02**(08), 6 (2010)
49. Zhao, Y., Wu, J., Li, F., Lu, S.: On maximizing the lifetime of wireless sensor networks using virtual backbone scheduling. IEEE Trans. Parallel Distrib. Syst. **23**(8), 1528–1535 (2012)

Chapter 26
Security Testing of Internet of Things for Smart City Applications: A Formal Approach

Moez Krichen, Mariam Lahami, Omar Cheikhrouhou, Roobaea Alroobaea, and Afef Jmal Maâlej

26.1 Introduction

Internet of Things (IoT) is a promising technology that permits to connect everyday things or objects to the Internet by giving them the capabilities to sense the environment and interact with other objects and/or human beings through the Internet. This evolving technology has promoted a new generation of innovative and valuable services. Today cities are getting smarter by deploying intelligent systems for traffic control, water management, energy management, public transport, street lighting, etc., thanks to these services. Nevertheless, these services can easily be compromised and attacked by malicious parties in the absence of proper mechanism for providing adequate security.

Recent studies have shown that the attackers are using smart home appliances to launch serious attacks such as infiltrating to the network or sending malicious email or launching malicious actions such as distributed denial of service (DDoS) attack. Therefore, security solutions need to be proposed, set up, and tested to mitigate these identified attacks.

M. Krichen (✉)
Faculty of CSIT, Al-Baha University, Al Baha, Saudi Arabia

ReDCAD Laboratory, University of Sfax, Sfax, Tunisia
e-mail: moez.krichen@redcad.org

M. Lahami · A. J. Maâlej
ReDCAD Laboratory, University of Sfax, Sfax, Tunisia
e-mail: mariam.lahami@redcad.org; afef.jmal@redcad.org

O. Cheikhrouhou · R. Alroobaea
Taif University, Taif, Saudi Arabia
e-mail: o.cheikhrouhou@tu.edu.sa; r.robai@tu.edu.sa

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_26

In this work, we aim to adopt a model-based security testing (MBST) approach to check the security of IoT applications in the context of smart cities. The MBST approach consists in specifying the desired IoT application in an abstract manner using an adequate formal specification language and then deriving test suites from this specification to find security vulnerabilities in the application under test in a systematic manner.

The work introduced here is an extension of a previous work [19] and it is a piece of a broader approach dealing with the security of IoT applications for smart cities and consisting of the following steps:

- Identify and assess the threats and the attacks in smart cities IoT applications.
- Design and develop security mechanisms for standard protocols at the application and the network layer.
- Evaluate the performance and the correctness of the proposed security protocols using simulation and implementation on real devices.

The rest of this paper is organized as follows. Section 26.2 introduces some preliminaries about IoT and smart cities. Section 26.3 defines the model of extended timed automata. Section 26.4 presents our conformance testing framework. Section 26.5 presents an overview about our proposed approach. Section 26.6 reports on related research efforts dealing with IoT security testing. Finally Sect. 26.7 concludes the paper.

26.2 Preliminaries

26.2.1 Internet of Objects

Recent advances in communication and sensing devices make our everyday objects smarter. This smartness is resulted from the capability of objects to sense the environment, to process the captured (sensed) data, and to communicate it to users either directly or through the Internet. Taking an example of the object "lamp," a classical lamp needs to be wired, linked to the electricity in order to produce light and it does not handle more than the on and off states. This lamp becomes smarter if it is equipped with sensors that can detect environment luminosity and adjust its brightness automatically based on the sensed value. Moreover, this lamp can be equipped with a communication system and therefore can be remotely controlled and supervised (e.g., energy consumption). This example can be generalized to any other thing (object) and therefore leading to the Internet of Things (IoT) concept. The IoT refers to the ability of everyday objects to connect to the Internet and to send and receive data. The integration of these smart objects to the Internet infrastructure is promoting a new generation of innovative and valuable services for people. These services include home automation, traffic control, public transportation, smart water

metering, waste and energy management, etc. When integrated in a city context, they make citizens live better and so form the modern smart city.

26.2.2 Smart Cities

In the recent years, several research works are shaping the smart cities concept [4, 34]. In October 2015, ITU-T's Focus Group on Smart Sustainable Cities (FG-SSC)[1] agreed on the following definition of a smart sustainable city: A smart sustainable city (SSC) is an innovative city that uses information and communication technologies (ICTs) and other means to improve quality of life, efficiency of urban operation and services, and competitiveness, while ensuring that it meets the needs of present and future generations with respect to economic, social, and environmental aspects. Based on this definition, the main goal for SSC is to enhance the quality of life of its citizens across multiple, interrelated dimensions, including (but not limited to) the provision and access to water resources, energy, transportation and mobility, education, environment, waste management, housing, and livelihoods (e.g., jobs), utilizing ICTs as the key medium. Therefore, the IoT as a promising ICT technology will play a major role in the development of these new smart cities. With IoT, objects like phones, cars, household appliances, or clothes become wirelessly connected and can sense and share data.

26.2.3 Threats

Indeed, connecting our everyday things to the public Internet opens these objects to several kinds of attacks. Take the example of a traffic control system. If the hackers could insert fake messages to these traffic control system devices, they can make traffic perturbations and bottlenecks. Another example related to home automation, if attackers gain access to smart devices such as lamps and doors, it could manipulate doors and steal the house properties. The main security threats in the IoT are summarized in [8] as follows:

- Cloning of smart things by untrusted manufacturers;
- Malicious substitution of smart things during installation;
- Firmware replacement attack;
- Extraction of security parameters since smart things may be physically unprotected;
- Eavesdropping attack if the communication channel is not adequately protected;
- Man-in-the-middle attack during key exchange;

[1] https://www.itu.int/en/ITU-T/focusgroups/ssc.

– Routing attacks;
– Denial-of-service attacks; and
– Privacy threats.

Therefore, a key challenge for IoT towards smart city applications is ensuring its reliability, security, and privacy, which represent the main scope of this work. Without guarantees that smart city IoT objects are: (1) sensing correctly the environment, (2) exchanging the information securely, (3) safeguarding private information, and (4) providing correct and not manipulated information, users are reluctant to adopt this new technology that will be a part of their everyday lives.

26.2.4 Challenges

Due to its specific characteristic, new issues are raised in the area of IoT. Trust management, which plays an important role in the IoT for reliable data fusion, qualified services, and enhanced user privacy and information security, is one of these main issues. Indeed, data collection trust is a crucial issue in the IoT. If the huge collected data is not trusted (e.g., due to the damage or malicious input of some sensors), the IoT service quality will be greatly influenced and hard to be accepted by users even though the network layer trust and the application layer trust can be fully provided [36]. On the other hand, in order to have intelligent context-aware services, users should disclose or have to share their personal data or privacy such as location, preferences, and contacts. Providing intelligent context-aware and personalized services and at the same time preserving user privacy are two conflicting objectives that induce a big challenges in the IoT. Another challenge faced when designing security solutions to the IoT is the limited resources of the IoT devices. Indeed, most of IoT devices are limited in terms of CPU, memory capacity, and battery supply. They often operate on lossy and low bandwidth communication channels. This renders the application of the conventional Internet security solutions not appropriate. As an example, the use of small packets (i.e., IEEE 802.15.4 supports only 127-bytes packets) may result in fragmentation of larger packets when using the standard protocols. This will quickly exhaust the lifetime of IoT devices and open new possibilities for DoS attacks [25]. Moreover, the limited resources of IoT devices render the use of complex cryptographic protocols inadequate. Finally, the inherent complexity of the IoT, where multiple heterogeneous entities located in different contexts can exchange information with each other, further complicates the design and the deployment of efficient, interoperable, and scalable security mechanisms [28]. The proposed security solutions should fulfill the following security requirements [26].

– Data confidentiality: make information inaccessible to unauthorized users. For example, a 6LoWPAN node should not leak some of its collected data to neighboring networks.

- Data authentication: since an adversary can easily inject messages, the receiver needs to ensure that data are originated from trusted sources.
- Data integrity: ensures that an adversary does not alter the received data in transit.
- Availability: ensures the survivability of network services to (only) authorized parties when needed, despite a DoS attack(s).
- Energy efficiency: a security scheme must be energy efficient so as to maximize the network lifetime.

26.3 Extended Timed Automata

We extend the framework presented in [18].

26.3.1 Timed Labeled Transition Systems

Let R be the set of non-negative reals, Q the set of non-negative rationals, and N the set of non-negative integers. Given a finite set of *actions* Ac, the set $(\mathsf{Ac} \cup \mathsf{R})^*$ of all finite-length *real-time sequences* over Ac will be denoted $\mathsf{RT}(\mathsf{Ac})$. $\epsilon \in \mathsf{RT}(\mathsf{Ac})$ is the empty sequence. Given $\mathsf{Ac}' \subseteq \mathsf{Ac}$ and $\rho \in \mathsf{RT}(\mathsf{Ac})$, $\mathsf{P}_{\mathsf{Ac}'}(\rho)$ denotes the *projection* of ρ to $\mathsf{Ac}' \cup \mathsf{R}$, obtained by "erasing" from ρ all actions not in $\mathsf{Ac}' \cup \mathsf{R}$. Similarly, $\mathsf{DP}_{\mathsf{Ac}'}(\rho)$ denotes the (discrete) projection of ρ to Ac'. For example, if $\mathsf{Ac} = \{a, b\}$, $\mathsf{Ac}' = \{a\}$ and $\rho = a\,1\,b\,2\,a\,3$, then $\mathsf{P}_{\mathsf{Ac}'}(\rho) = a\,3\,a\,3$ and $\mathsf{DP}_{\mathsf{Ac}'}(\rho) = a\,a$. The time spent in a sequence ρ, denoted $\mathsf{duration}(\rho)$, is the sum of all delays in ρ, for example, $\mathsf{duration}(\epsilon) = 0$ and $\mathsf{duration}(a\,1\,b\,0.5) = 1.5$. In the rest of the document, we assume given a set of actions Ac, partitioned in two disjoint sets: a set of *input actions* $\mathsf{Ac}_{\mathsf{in}}$ and a set of *output actions* $\mathsf{Ac}_{\mathsf{out}}$. Actions in $\mathsf{Ac}_{\mathsf{in}} \cup \mathsf{Ac}_{\mathsf{out}}$ are called *observable* actions. We also assume there is an *unobservable* action $\tau \notin \mathsf{Ac}$. Let $\mathsf{Ac}_\tau = \mathsf{Ac} \cup \{\tau\}$. A *timed labeled transition system* (TLTS) over Ac is a tuple $(S, s_0, \mathsf{Ac}, T_d, T_t)$, where:

- S is a set of *states*;
- s_0 is the initial state; T_d is a set of *discrete transitions* of the form (s, a, s') where $s, s' \in S$ and $a \in \mathsf{Ac}$;
- T_t is a set of *timed transitions* of the form (s, t, s') where $s, s' \in S$ and $t \in \mathsf{R}$.

Timed transitions must be deterministic, that is, $(s, t, s') \in T_t$ and $(s, t, s'') \in T_t$ implies $s' = s''$. T_t must also satisfy the following conditions: $(s, t, s') \in T_t$ and $(s', t', s'') \in T_t$ implies $(s, t + t', s'') \in T_t$; $(s, t, s') \in T_t$ implies that for all $t' < t$, there is some $(s, t', s'') \in T_t$.

We use standard notation concerning TLTS. For $s, s', s_i \in S$, $\mu, \mu_i \in \mathsf{Ac}_\tau \cup \mathsf{R}$, $a, a_i \in \mathsf{Ac} \cup \mathsf{R}$, $\rho \in \mathsf{RT}(\mathsf{Ac}_\tau)$ and $\sigma \in \mathsf{RT}(\mathsf{Ac})$, we have

- General transitions:
  - $s \xrightarrow{\mu} s' \stackrel{Def}{=} (s, \mu, s') \in T_d \cup T_t$; $s \xrightarrow{\mu} \stackrel{Def}{=} \exists s' : s \xrightarrow{\mu} s'$;
  - $s \not\xrightarrow{\mu} \stackrel{Def}{=} \nexists s' : s \xrightarrow{\mu} s'$;
  - $s \xrightarrow{\mu_1 \cdots \mu_n} s' \stackrel{Def}{=} \exists s_1, \cdots, s_n : s = s_1 \xrightarrow{\mu_1} s_2 \xrightarrow{\mu_2} \cdots \xrightarrow{\mu_n} s_n = s'$;
  - $s \xrightarrow{\rho} \stackrel{Def}{=} \exists s' : s \xrightarrow{\rho} s'$;
  - $s \not\xrightarrow{\rho} \stackrel{Def}{=} \nexists s' : s \xrightarrow{\rho} s'$.
- Observable transitions:
  - $s \stackrel{\epsilon}{\Rightarrow} s' \stackrel{Def}{=} s = s'$ or $s \xrightarrow{\tau \cdots \tau} s'$;
  - $s \stackrel{a}{\Rightarrow} s' \stackrel{Def}{=} \exists s_1, s_2 : s \stackrel{\epsilon}{\Rightarrow} s_1 \xrightarrow{a} s_2 \stackrel{\epsilon}{\Rightarrow} s'$;
  - $s \stackrel{a}{\not\Rightarrow} \stackrel{Def}{=} \nexists s' : s \stackrel{a}{\Rightarrow} s'$; $s \stackrel{a_1 \cdots a_n}{\Longrightarrow} s' \stackrel{Def}{=} \exists s_1, \cdots, s_n : s = s_1 \stackrel{a_1}{\Rightarrow} s_2 \stackrel{a_2}{\Rightarrow} \cdots \stackrel{a_n}{\Rightarrow} s_n = s'$;
  - $s \stackrel{\sigma}{\Rightarrow} \stackrel{Def}{=} \exists s' : s \stackrel{\sigma}{\Rightarrow} s'$;
  - $s \stackrel{\sigma}{\not\Rightarrow} \stackrel{Def}{=} \nexists s' : s \stackrel{\sigma}{\Rightarrow} s'$.

A sequence of the form $s_0 \xrightarrow{\mu_1} s \xrightarrow{\mu_2} \cdots \xrightarrow{\mu_n} s'$ is called a *run* and a sequence of the form $s_0 \stackrel{a_1}{\Rightarrow} s \stackrel{a_2}{\Rightarrow} \cdots \stackrel{a_n}{\Rightarrow} s'$ an *observable run*.

26.3.2 Extended Timed Automata

We use timed automata [2] with deadlines to model urgency [18]. An *extended timed automaton over* Ac is a tuple $A = (Q, q_0, X, \mathsf{I}, \mathsf{Ac}, \mathsf{E})$, where:

- Q is a finite set of *locations*;
- $q_0 \in Q$ is the initial location;
- X is a finite set of *clocks*;
- I is a finite set of integer variables;
- E is a finite set of *edges*.

Each edge is a tuple $(q, q', \psi, r, inc, dec, d, a)$, where:

- $q, q' \in Q$ are the source and destination locations;
- ψ is the *guard*, a conjunction of constraints of the form $x\#c$, where $x \in X \cup \mathsf{I}$, c is an integer constant and $\# \in \{<, \leq, =, \geq, >\}$;
- $r \subseteq X \cup \mathsf{I}$ is a set of clocks and integer variables to *reset* to zero;
- $inc \subseteq \mathsf{I}$ is a set of integer variables (disjoint from r) to *increment* by one;
- $dec \subseteq \mathsf{I}$ is a set of integer variables (disjoint from r and inc) to *decrement* by one;
- $d \in \{\mathsf{lazy}, \mathsf{delayable}, \mathsf{eager}\}$ is the *deadline*;
- $a \in \mathsf{Ac}$ is the action.

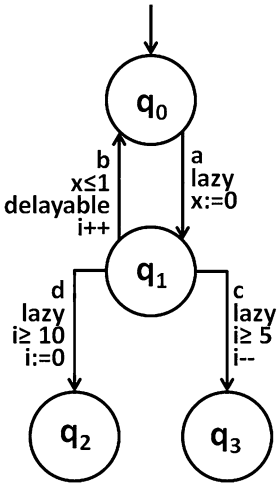


Fig. 26.1 An example of an extended timed automaton

An example of an extended timed automaton $A = (Q, q_0, X, \mathsf{I}, \mathsf{Ac}, \mathsf{E})$ over the set of actions $\mathsf{Ac} = \{a, b, c, d\}$ is given in Fig. 26.1 where:

- $Q = \{q_0, q_1, q_2, q_3\}$ is the set of locations;
- q_0 is the initial location;
- $X = \{x\}$ is the finite set of clocks;
- $\mathsf{I} = \{i\}$ is the finite set of integer variables;
- E is the set of edges drawn in the figure.

The figure uses the following notation:

- "$x := 0$" means resetting the clock x to 0;
- "$i := 0$" means resetting the integer variable i to 0;
- "$i + +$" means incrementing i by 1;
- "$i - -$" means decrementing i by 1.

26.3.3 Semantics of Extended Timed Automata

An extended timed automaton $A = (Q, q_0, X, \mathsf{I}, \mathsf{Ac}, \mathsf{E})$ defines an infinite TLTS which is denoted $L_A = (S_A, s_0^A, \mathsf{Ac}, T_d^A, T_t^A)$.

- Its states S_A are tuples $s = (q, v_X, v_\mathsf{I})$, where:
  - $q \in Q$;
  - $v_X : X \to \mathsf{R}$ is a *clock valuation*;
  - and $v_\mathsf{I} : \mathsf{I} \to \mathsf{N}$ is a *integer variable valuation*.

- $s_0^A = (q_0, \mathbf{0}_X, \mathbf{0}_I)$ is the initial state, where:
  - $\mathbf{0}_X$ is the valuation assigning 0 to every clock of A;
  - $\mathbf{0}_I$ is the valuation assigning 0 to every integer variable of A.
- *Discrete transitions* are of the form $(q, v_X, v_I) \xrightarrow{a} (q', v'_X, v'_I)$ where $a \in \mathsf{Ac}$ and there is an edge $(q, q', \psi, r, inc, dec, d, a)$ such that (v_X, v_I) satisfies ψ and (v'_X, v'_I) is obtained by:
  - resetting to zero all clocks and integer variables in r;
  - incrementing integer variables in *inc* by one;
  - decrementing variables in *dec* by one;
  - leaving all other variables unchanged.
- *Timed transitions* are of the form $(q, v_X, v_I) \xrightarrow{t} (q, v_X + t, v_I)$ where $t \in \mathsf{R}, t > 0$ and there is no edge $(q, q'', \psi, r, inc, dec, d, a)$ such that:
  - either $d = \mathsf{delayable}$ and there exist $0 \leq t_1 < t_2 \leq t$ such that $(v_X + t_1, v_I) \models \psi$ and $(v_X + t_2, v_I) \not\models \psi$;
  - or $d = \mathsf{eager}$ and $(v_X, v_I) \models \psi$.

Lazy edges do not impact the semantics. They denote that an edge is neither delayable nor eager. More precisely, lazy edges cannot block time progress, whereas delayable and eager edges can. We do not allow delayable edges with guards of the form $x < c$ since there is no *latest* time when the guard is still true. Similarly, we do not allow eager edges with guards of the form $x > c$ since there is no *earliest* time when the guard becomes true.

A state $s \in S_A$ is *reachable* if there exists $\rho \in \mathsf{RT}(\mathsf{Ac})$ such that $s_0^A \xrightarrow{\rho} s$. The set of reachable states of A is denoted $\mathsf{Reach}(A)$.

For instance, for the TA presented in Fig. 26.1, the initial state is $(q_0, 0, 0)$. A possible timed transition of the system is $(q_0, 0, 0) \xrightarrow{5} (q_0, 5, 0)$ which corresponds to the fact that the system spends 5 time units at node q_0. A possible discrete transition is $(q_0, 5, 0) \xrightarrow{a} (q_1, 0, 0)$ which corresponds to the execution of action a and results in the reset of clock x to 0. Another possible discrete transition is $(q_1, 0, 0) \xrightarrow{b} (q_0, 0, 1)$ by which the integer variable i is incremented for the first time. The time constraint $x \leq 1$ means that the execution of b must happen at most 1 time unit after the execution of a. The deadline delayable associated with this constraint means that time is blocked after one time unit and that it is compulsory to execute action b before that limit. It is not difficult to see that action b needs to be executed at least 5 times (resp., 10 times) in order to execute action c (resp., d).

26.3.4 Extended Timed Automata with Inputs and Outputs

An *extended timed automaton with inputs and outputs* (ETAIO) is an extended timed automaton over the partitioned set of actions $\mathsf{Ac}_\tau = \mathsf{Ac}_{\mathsf{in}} \cup \mathsf{Ac}_{\mathsf{out}} \cup \{\tau\}$. For clarity, we will explicitly include inputs and outputs in the definition of an ETAIO A and write $(Q, q_0, X, \mathsf{I}, \mathsf{Ac}_{\mathsf{in}}, \mathsf{Ac}_{\mathsf{out}}, \mathsf{E})$ instead of $(Q, q_0, X, \mathsf{I}, \mathsf{Ac}_\tau, \mathsf{E})$.

- An ETAIO is called *observable* if none of its edges is labeled by τ.
- Given a set of inputs $\mathsf{Ac}' \subseteq \mathsf{Ac}_{\mathsf{in}}$, an ETAIO A is called *input-enabled* with respect to Ac' if it can accept any input in Ac' at any state:

$$\forall s \in \mathsf{Reach}(A) \,.\, \forall a \in \mathsf{Ac}' \,:\, s \xrightarrow{a} .$$

  It is simply said to be input-enabled when $\mathsf{Ac}' = \mathsf{Ac}_{\mathsf{in}}$.
- A is called lazy-*input* with respect to Ac' if the deadlines on all the transitions labeled with input actions in Ac' are lazy. It is called lazy-input if it is lazy-input with respect to $\mathsf{Ac}_{\mathsf{in}}$. Note that input-enabled does not imply lazy-input in general.
- A is called *deterministic* if:

$$\forall s, s', s'' \in \mathsf{Reach}(A) \,.\, \forall a \in \mathsf{Ac}_\tau \,:\, s \xrightarrow{a} s' \wedge s \xrightarrow{a} s'' \Rightarrow s' = s''.$$

- A is called *non-blocking* if:

$$\forall s \in \mathsf{Reach}(A) \,.\, \forall t \in \mathsf{R} \,.\, \exists \rho \in \mathsf{RT}(\mathsf{Ac}_{\mathsf{out}} \cup \{\tau\}) \,:\, \mathsf{duration}(\rho) = t \wedge s \xrightarrow{\rho} .$$

  This condition guarantees that A will not block time in any environment.

  The set of *timed traces* of an ETAIO A is defined to be

$$\mathsf{TTr}(A) = \{\rho \mid \rho \in \mathsf{RT}(\mathsf{Ac}_\tau) \wedge s_0^A \xrightarrow{\rho}\}.$$

  The set of *observable timed traces* of A is defined to be

$$\mathsf{OTTr}(A) = \{\mathsf{P}_{\mathsf{Ac}}(\rho) \mid \rho \in \mathsf{RT}(\mathsf{Ac}_\tau) \wedge s_0^A \xrightarrow{\rho}\}.$$

The TLTS defined by an ETAIO is called a *timed input–output LTS* (TIOLTS). From now on, unless otherwise stated, all the considered ETAIO are defined with respect to the same sets $\mathsf{Ac}_{\mathsf{in}}$ and $\mathsf{Ac}_{\mathsf{out}}$ and unobservable action τ. As for ETAIO, a given TIOLTS L is denoted $(S, s_0, \mathsf{Ac}_{\mathsf{in}}, \mathsf{Ac}_{\mathsf{out}}, T_d, T_t)$ instead of $(S, s_0, \mathsf{Ac}_\tau, T_d, T_t)$. The two operators $\mathsf{TTr}(\cdot)$ and $\mathsf{OTTr}(\cdot)$ are extended in a natural way to the case of TIOLTS.

26.3.5 Parallel Composition of ETAIO with Shared Integer Variables

The parallel composition we propose here is similar to the parallel composition for classical timed automata. The new thing here is that we consider shared variables between the different elements to compose. The shared variables can be incremented and decremented by any participant in the composition. These variables are used to formulate the constraints of the different automata. In this way the behaviors of the different components of the system are related to each other and depend on each other. For instance, the shared variables may represent the shared resources of the system.

Let n be a non-negative integer such that $n \geq 2$. We consider n ETAIO $(A_i)_{1 \leq i \leq n}$ where $A_i = (Q^i, q_0^i, X^i, \mathsf{I}, \mathsf{Ac}_{\mathsf{in}}^i, \mathsf{Ac}_{\mathsf{out}}^i, \mathsf{E}^i)$. That is the set of integer variables I is shared between all the considered ETAIO $(A_i)_{1 \leq i \leq n}$ while no other element from Q^i, X^i, $\mathsf{Ac}_{\mathsf{in}}^i$, and $\mathsf{Ac}_{\mathsf{out}}^i$ is shared with the other ETAIO $(A_j)_{j \neq i}$.

The TIOLTS $L^P = (S^P, s_0^P, \mathsf{Ac}_{\mathsf{in}}^P, \mathsf{Ac}_{\mathsf{out}}^P, T_d^P, T_t^P)$ generated by the parallel product of the ETAIO $(A_i)_{1 \leq i \leq n}$ is defined as follows:

- $s_0^P = ((q_0^1, \cdots, q_0^n), (\mathbf{0}_{X^0}, \cdots, \mathbf{0}_{X^n}), \mathbf{0}_{\mathsf{I}})$;
- $\mathsf{Ac}_{\mathsf{in}}^P = \bigcup_{1 \leq i \leq n} \mathsf{Ac}_{\mathsf{in}}^i$, $\mathsf{Ac}_{\mathsf{out}}^P = \bigcup_{1 \leq i \leq n} \mathsf{Ac}_{\mathsf{out}}^i$;
- and S^P, T_d^P, and T_t^P are the smallest sets such that
  - $s_0^P \in S^P$;
  - For $s^P = ((q^1, \cdots, q^n), (v_{X^0}, \cdots, v_{X^n}), v_{\mathsf{I}}) \in S^P$ and $\delta \in \mathsf{R}$:

$$\forall 1 \leq i \leq n : (q_i, v_{X^i}, v_I) \xrightarrow{\delta} (q_i, v_{X^i} + \delta, v_I) \in T_t^i$$

$$\Rightarrow s'^P = ((q^1, \cdots, q^n), (v_{X^0} + \delta, \cdots, v_{X^n} + \delta), v_{\mathsf{I}}) \in S^P \; and \; s^P \xrightarrow{\delta} s'^P \in T_t.$$

  - For $s^P = ((q^1, \cdots, q^n), (v_{X^0}, \cdots, v_{X^n}), v_{\mathsf{I}}) \in S^P$, $1 \leq i \leq n$ and $a_i \in \mathsf{Ac}_\tau{}^i = \mathsf{Ac}_{\mathsf{in}}^i \cup \mathsf{Ac}_{\mathsf{out}}^i \cup \{\tau\}$):

$$(q_i, v_{X^i}, v_I) \xrightarrow{a_i} (q_i', v_{X^i}', v_I') \in T_d^i$$

$$\Rightarrow s'^P = (q'^p, v_X'^p, v_{\mathsf{I}}') \in S^P \quad \wedge \quad s^P \xrightarrow{a_i} s'^P \in T_d$$

where

$$q'^p = (q^1, \cdots, q^{i-1}, q'^i, q^{i+1}, \cdots, q^n)$$

and

$$v_X'^p = (v_{X^0}, \cdots, v_{X^{i-1}}, v_{X^i}', v_{X^{i+1}} \cdots, v_{X^n}).$$

It is worth noticing here that it is possible to define the parallel composition of n copies $(A_i)_{1 \leq i \leq n}$ of the same ETAIO A. In this case we assume it is possible to distinguish the sets of inputs and outputs of the different instances by a particular identifier corresponding to each instance. Obviously, the n instances share the set of integer variables of the ETAIO A. The obtained TIOLTS is denoted L_n^P.

26.4 Conformance Testing Framework

In this section, we are going to define a new extended timed input–output conformance relation, etioco. Then, we propose a new approach for deriving analog-clock tests from the SUT specification. Finally, we discuss both test execution and correctness requirements.

26.4.1 Conformance Relation

In order to formally define the conformance relation, we define a number of operators. Given a TIOLTS $L = (S^L, s_0^L, \mathsf{Ac}_{\mathsf{in}}^L, \mathsf{Ac}_{\mathsf{out}}^L, T_d^L, T_t^L)$ and a timed trace $\sigma \in \mathsf{RT}(\mathsf{Ac}^L)$ L after σ is the set of all states of L that can be reached by some timed sequence ρ whose projection to observable actions is σ. Formally: L after $\sigma = \{s \in S^L \mid \exists \rho \in \mathsf{RT}(\mathsf{Ac}_\tau{}^L) : s_0^L \xrightarrow{\rho} s \wedge \mathsf{P}_{\mathsf{Ac}}(\rho) = \sigma\}$. Given state $s \in S^L$, $\mathsf{elapse}(s)$ is the set of all delays which can elapse from s without L making any observable action. Formally: $\mathsf{elapse}(s) = \{t > 0 \mid \exists \rho \in \mathsf{RT}(\{\tau\}) : \mathsf{duration}(\rho) = t \wedge s \xrightarrow{\rho}\}$. Given state $s \in S^L$, $\mathsf{out}(s)$ is the set of all observable "events" (outputs or the passage of time) that can occur when the system is at state s. The definition naturally extends to a set of states S. Formally: $\mathsf{out}(s) = \{a \in \mathsf{Ac}_{\mathsf{out}}^L \mid s \xrightarrow{a}\} \cup \mathsf{elapse}(s)$ and

$$\mathsf{out}(S) = \bigcup_{s \in S} \mathsf{out}(s).$$

The specification of the system to be tested is given as a non-blocking ETAIO A_S while the implementation can be modeled as a non-blocking, input-enabled ETAIO A_I. For $n \geq 1$, let $L_{S,n}^P$ (resp., $L_{I,n}^P$) be the parallel composition of n copies of A_S (resp., A_I). Input-enabledness is required so that the implementation can accept inputs from the tester at any state. The *extended timed input–output conformance relation*, denoted etioco, is an extension of our previous conformance relation tioco [17, 18]. The new relation etioco is defined as A_I etioco A_S *iff* $\forall n \geq 1 \wedge \sigma \in \mathsf{OTTr}(L_{S,n}^P) : \mathsf{out}(L_{I,n}^P \ \mathsf{after}\ \sigma) \subseteq \mathsf{out}(L_{S,n}^P \ \mathsf{after}\ \sigma)$. The relation states that an implementation A_I conforms to a specification A_S iff for any number of copies n of A_S and any observable behavior σ of $L_{S,n}^P$, the set of observable outputs

of $L^P_{I,n}$ after any behavior "matching" σ must be a subset of the set of possible observable outputs of $L^P_{S,n}$. Notice that observable outputs are not only observable output actions but also time delays. Also notice that in case we consider only $n = 1$, the definitions of etioco and tioco become the same.

26.4.2 Analog-Clock Tests

A test (or *test case*) is an experiment performed on the implementation by an agent (the *tester*). There are different types of tests, depending on the capabilities of the tester to observe and react to events. In general, one may consider either *analog-clock* or *digital-clock* tests [11]. In this work, we consider only analog-clock tests. The latter can measure precisely the delay between two observed actions and can emit an input at any point in time.

It should be noted that we consider *adaptive* tests (following the terminology of [21]), where the action the tester takes depends on the observation history. For $n \geq 1$, let Ac^n (resp., $\mathsf{Ac}^n_{\mathsf{in}}$) denote the union of all observable actions (resp., all input actions) of n copies of the specification A_S. An analog-clock test for n parallel executions of A_S is a total function $T_n : \mathsf{RT}(\mathsf{Ac}^n) \rightarrow \mathsf{Ac}^n_{\mathsf{in}} \cup \{\mathsf{Wait}, \mathsf{Pass}, \mathsf{Fail}\}$. $T_n(\rho)$ specifies the action the tester must take once it observes ρ:

- If $T_n(\rho) = a \in \mathsf{Ac}^n_{\mathsf{in}}$, then the tester emits input a.
- If $T_n(\rho) = \mathsf{Wait}$, then the tester waits (lets time elapse).
- If $T_n(\rho) \in \{\mathsf{Pass}, \mathsf{Fail}\}$, then the tester produces a verdict (and stops).

26.4.3 Test Execution and Correctness Requirements

The execution of the test T_n on the implementation A_I can be defined as the *parallel composition* of the TIOLTS defined by T_n and $L^P_{I,n}$ the TIOLTS corresponding to n copies of A_I, with the usual *synchronization* rules for transitions carrying the same label. We will denote the product TIOLTS by $L^P_{I,n} \| T_n$. The execution of the test reaches a pass/fail verdict after bounded time. Formally, we say that A_I *passes* the test, denoted A_I passes T_n, if state Fail is not reachable in the product $L^P_{I,n} \| T_n$. We say that an implementation passes (resp. fails) a set of tests (or *test suite*) $\mathcal{T}$ if it passes all tests (resp. fails at least one test) in $\mathcal{T}$. We say that an analog-clock test suite $\mathcal{T}$ is *sound with respect to* A_S if

$$\forall A_I \ : \ A_I \text{ etioco } A_S \Rightarrow A_I \text{ passes } \mathcal{T}.$$

We say that $\mathcal{T}$ is *complete with respect to* A_S if

$$\forall A_I \ : \ A_I \text{ passes } \mathcal{T} \Rightarrow A_I \text{ etioco } A_S.$$

26.5 Proposed Approach

In this section, we define a workflow that covers the different steps of a classical model-based testing process, namely: model Specification, test generation, test selection, test execution, and evaluation activities as depicted in Fig. 26.2.

26.5.1 Test Generation and Selection

Test Generation We adapt the untimed test generation algorithm of [30]. Roughly speaking, the algorithm builds a test in the form of a tree. A node in the tree is a set of states S of the specification and represents the "knowledge" of the tester at the current test state. The algorithm extends the test by adding successors to a leaf node, as illustrated in Fig. 26.3. For all *illegal* outputs a_i (outputs which cannot occur from any state in S) the test leads to Fail. For each legal output b_i, the test proceeds to node S_i, which is the set of states the specification can be in after emitting b_i (and possibly performing unobservable actions). If there exists an input c which can be accepted by the specification at some state in S, then the test may decide to emit this input (dashed arrow from S to S'). At any node, the algorithm may decide to stop the test and label this node as Pass.

Analog-clock tests cannot be directly represented as a finite tree, because there is an a-priori infinite set of possible observable delays at a given node. To remedy this, we use the idea of [31]. We represent an analog-clock test as an *algorithm*. The latter essentially performs subset construction on the specification automaton, during the

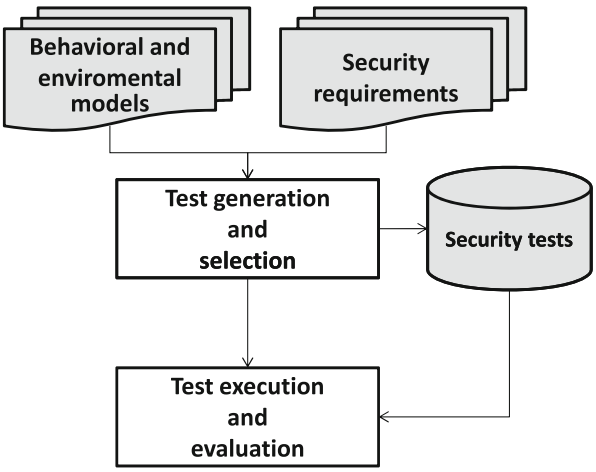


Fig. 26.2 Model-based security testing process

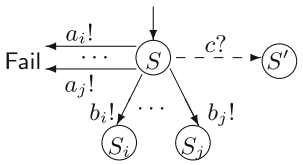


Fig. 26.3 Test generation principle [18]

Algorithm 1 On-the-fly analog-clock test generation

```
1  S ← tsucc({s^P_{n,0}}, 0);
2  while(not Fail)
3    x ← 0; /* x is a clock measuring elapsing time */
4    await(output b is received at x < T or x = T)
5    if (b received at x)
6      S ← dsucc(tsucc(S, x), b);
7    else
8      S ← tsucc(S, T);
9    endif;
10   if (S = ∅)
11     announce Fail;
12     exit ;
13   endif;
14   if (validinputs(S) ≠ ∅)
15     i ← pick({0, 1}); /* 0 to send an input and 1 to continue observation */
16   endif;
17   if (i = 0)
18     a ← pick(validinputs(S));
19     S ← dsucc(S, a);
20   endif;
21 endwhile;
```

execution of the test. Thus, our analog-clock testing method can be classified as on-the-fly or *on-line*, meaning that the test is generated at the same time it is executed. More precisely, the tester will maintain a set of states S of the TIOLTS $L^P_{S,n}$.

S will be updated every time an action is observed or some time delay elapses. Since the time delay is not known a-priori, it must be an input to the update function. We define the following operators:

$$\mathsf{dsucc}(S, a) = \{s' \mid \exists s \in S \ : \ s \xrightarrow{a} s'\}$$

and

$$\mathsf{tsucc}(S, t) = \{s' \mid \exists s \in S \,.\, \exists \rho \in \mathsf{RT}(\{\tau\}) \ : \ \mathsf{duration}(\rho) = t \wedge s \xrightarrow{\rho} s'\}$$

where $a \in \mathsf{Ac}^n$ and $t \in \mathsf{R}$. $\mathsf{dsucc}(S, a)$ contains all states which can be reached by some state in S performing action a. $\mathsf{tsucc}(S, t)$ contains all states which can be reached by some state in S via a sequence ρ which contains no observable actions and takes exactly t time units. The test operates as follows. It starts at state $S_0 = \mathsf{tsucc}(\{s^P_{n,0}\}, 0)$ where $s^P_{n,0}$ is the initial state of $L^P_{S,n}$. Given current state S:

- if output a is received t time units after entering S, then S is updated to $\mathsf{dsucc}(\mathsf{tsucc}(S, t), a)$.
- If ever the set S becomes empty, the test announces Fail.
- At any point, for an input b, if $\mathsf{dsucc}(S, b) \neq \emptyset$, the test may decide to emit b and update its state accordingly.

On-line analog-clock test generation is performed by Algorithm 1. The algorithm keeps running as long as no non-conformance is detected. At any time the tester can stop testing and declare Pass. The algorithm uses the following notation. Given a nonempty set X, $\mathsf{pick}(X)$ chooses randomly an element in X.

Given a set of states S, $\mathsf{validinputs}(S)$ is defined as the set of valid inputs at S, that is: $\mathsf{validinputs}(S) = \{a \in \mathsf{Ac}^n_{\mathsf{in}} | \mathsf{dsucc}(\mathsf{tsucc}(S, 0), a) \neq \emptyset\}$. Following the same methodology as in [18] we can prove that the proposed test generation algorithm is both sound and complete. Indeed both frameworks and both approaches are based on IOLTS and at this level the algorithms are the same and the conformance relations are equivalent. The difference between the two frameworks is only at syntactic and structural levels. In [18] the authors consider only one instance of the system whereas in our case we consider many instances which interact with each other and the behaviors of which are influenced by the total number of active components and the state of the shared resources.

The used test generation technique is based on model checking. The main idea is to formulate the test generation problem as a reachability problem that can be solved with the model checker tool UPPAAL [3]. However, instead of using model annotations and reachability properties to express coverage criteria, the observer language is used.

In this direction, we reuse the finding of Hessel et al. [14] by exploiting its extension of UPPAAL, namely UPPAAL CO$\sqrt{}$ER.[2] This tool takes as inputs a model, an observer, and a configuration file. The model is specified as a network of timed automata (.xml) that comprises a SUT part and an environment part. The observer (.obs) expresses the coverage criterion that guides the model exploration during test case generation. The configuration file (.cfg) describes mainly the interactions between the system part and the environment part in terms of input/output signals. It may also specify the variables that should be passed as parameters in these signals. As output, it produces a test suite containing a set of timed traces (.xml).

[2] http://user.it.uu.se/~hessel/CoVer/index.php.

Our test generation module is built upon these well-elaborated tools. The key idea here is to use UPPAAL CO$\surd$ER and its generic and formal specification language for coverage criteria to generate tests for security purposes.

Test Selection Different coverage criteria have been proposed for software, such as statement coverage and branch coverage [24]. In the TA case existing methods attempt to cover either finite abstractions of the state space (e.g., the region graph [29]) or structural elements of the specification such as edges or locations [14]. Here, we propose a technique for covering states, locations, edges, actions, or shared variables of the specification:

- State coverage: As already mentioned each node of a given test case corresponds to a set of states S of A_S^{Tick}. We say that the node *covers* S. We say that such a test covers the union of all sets of states covered by its nodes. We say that a set of tests (or *test suite*) achieves *state coverage* if every reachable state of $L_{S,n}^P$ is covered by some test in the suite.
- Location coverage: A test suite achieves *location coverage* if every reachable location of A_S is covered by some test in the suite.
- Edge coverage: Every edge of a test case can be associated with an edge of $L_{S,n}^P$. In particular, an edge $S \xrightarrow{a} S'$ will be associated with all edges which are visited during the reachability algorithm which computes S' from S. We say that a test suite achieves *edge coverage* if every reachable edge of $L_{S,n}^P$ is covered by some test in the suite.
- Action coverage: We also define *action coverage* as follows. If a given edge $S \xrightarrow{a} S'$ is reachable, then the corresponding observable action a is said to be reachable as well. Action coverage is achieved if all the reachable observable actions are covered by the considered test suite.
- Shared integer variable coverage: Finally we define *shared integer variable coverage* which consists in generating tests which cover the different possible values of the system variables.

26.5.2 Test Execution and Verdict Analysis

For the execution of the obtained security tests, we aim to use a standard-based test execution platform, called TTCN-3 test system for runtime testing (TT4RT), developed in a previous work [20]. To do so, security tests should be mapped to the TTCN-3 notation since our platform supports only this test language. Then, test components are dynamically created and assigned to execution nodes in a distributed manner.

Each test component is responsible for (1) stimulating the SUT with input values, (2) comparing the obtained output data with the expected results (also called oracle), and (3) generating the final verdict. The latter can be pass, fail, or inconclusive. A pass verdict is obtained when the observed results are valid with respect to

the expected ones. A fail verdict is obtained when at least one of the observed results is invalid with respect to the expected one. Finally, an inconclusive verdict is obtained when neither a pass nor a fail verdict can be given. After computing for each executed test case its single verdict, the proposed platform deduces the global verdict.

26.5.3 Cloud Testing

The emergent paradigm, cloud computing, is formally defined by U.S.NIST (National Institute of Standards and Technology) [23] as follows. Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

This cloud model is characterized with three service models: software as-a-service (SaaS), platform as-a-service (PaaS), and infrastructure as-a-service (IaaS). The SaaS refers to the capability provided to the consumer to use the provider's applications running on a cloud infrastructure. With PaaS, the consumer is able to deploy his own applications without installing any platform or tools since he uses provided platform layer resources, including operating system support and software development frameworks. Regarding IaaS, it provides a collection of resources such as servers, storage, networks, and other computing resources in the form of virtualized systems, which are accessed through the Internet.

It is worthy to note that public cloud providers like Amazon Web Services[3] and Google Cloud Platform[4] offer a cloud infrastructure made up essentially of availability zones and regions. As shown in Fig. 26.4, a region is a specific geographical location in which public cloud service providers' data centers reside. Each region is further subdivided into availability zones. Several resources can live in a zone, such as instances or persistent disks. In the context of Google Cloud Platform, the us-central1 region, for example, denotes a region in the Central United States that has four zones, namely us-central1-a, us-central1-b, us-central1-c, and us-central1-f.

Cloud computing has been used in the context of software testing to encounter the lack of resources and the expensiveness of building a distributed test environment during the testing process. As a result, the concept of *cloud testing* is newly emerging in order to provide cost-effective testing services. According to [7], it refers to testing activities, essentially test generation, test execution, and test evaluation on a cloud-based environment. The latter supports on-demand resource allocation to large-scale testers whenever and wherever they need by following the

[3] https://aws.amazon.com/fr/.

[4] https://cloud.google.com/.

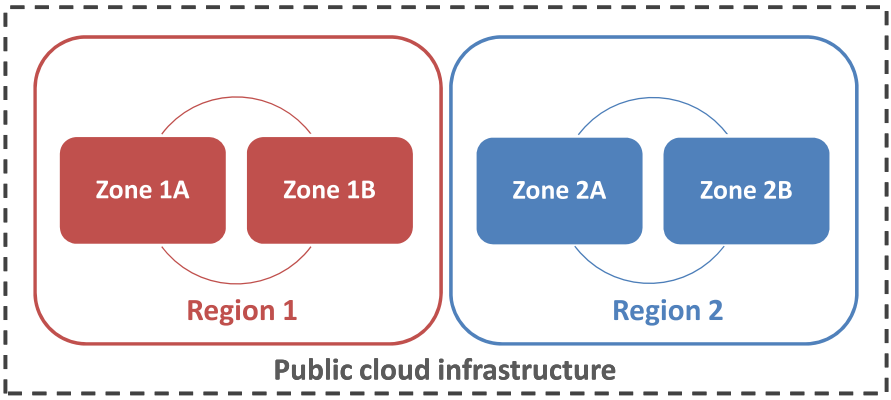


Fig. 26.4 Illustration of cloud partitioning in regions and zones

pay-per-use business model. Such virtualized and shared resources may reduce effectively the cost of building a distributed test environment for the runtime validation of dynamically adaptive systems.

Testing as-a-service (TaaS) is an innovative concept that provides end users with testing services such as test case generation, test execution, and test result evaluation. It has been proposed to improve the efficiency of software quality assurance. Notably, it is used for software systems that are remotely deployed in a virtualized runtime environment using shared hardware/software resources, and hosted in a third-party infrastructure (i.e., a cloud). One of the primary objectives is to reduce the cost of software testing tasks by providing on-demand testing services and also on-demand test environment services (i.e., establishing the required virtual (or physical) cloud-based computing resources and infrastructures for testing purposes).

26.5.4 Test Execution Platform as-a-Service

The proposed approach is built based on TaaS concepts. Figure 26.5 outlines an overview of its different constituents.

– Test management GUI: This component offers a graphical user interface (GUI) charged with managing the overall testing process: the automatic creation/deletion of VM instances, the dynamic allocation of test components to the appropriate VMs, the start-up of test component execution, and the computation of the final verdict. Moreover, it is responsible for querying the runtime monitoring component for information about the usage of resources in running VMs.

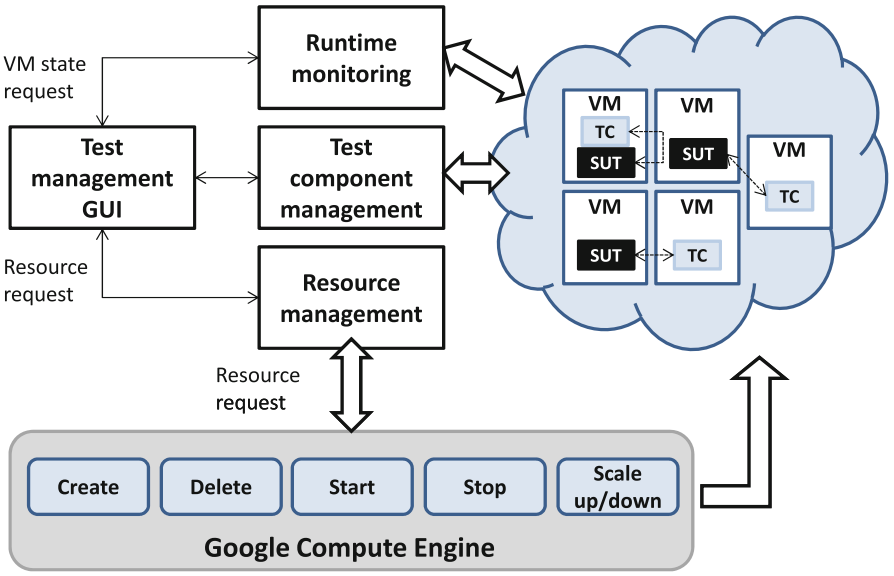


Fig. 26.5 Test execution platform overview

- Resource management: This component enables flexibility and elasticity during the testing process. If there is no adequate VM to handle the execution of a test component, a new VM can be created and started automatically. Moreover, it is possible to scale up or scale down an existing VM. The unused one can be released as well.
- Test component management: This component offers services for creating/deleting test components and starting/stopping their execution. A test component is an entity that interacts with the SUT to execute the available test cases (i.e., a set of input values and expected results) and to observe its response related to this excitation. Its main role consists of stimulating the SUT with the input values, comparing the obtained output values to the expected results and generating the final verdict that can be pass, fail, or inconclusive.
- Runtime monitoring: This component monitors VM instances during or even before the test execution and gives the status of each VM in terms of computing resources (such as CPU, memory, and storage).

As already discussed, several VM instances are created and started in the proposed cloud infrastructure in which several components under test are running too and can be evolved at runtime. To perform runtime tests in a cost-effective manner, several test components should be deployed in the final execution environment. The major question to be tackled here is how to assign efficiently test components to the existing VM instances?

Before answering to this question, we should mention that the components under test are distributed in several VM instances that can be located in the same region

and in the same zone as well as in different zones and even in different regions of the cloud infrastructure. Such information is provided via a SUT deployment descriptor. In this file, the SUT manager defines, for each component under test, the VM instance hosting its execution and also its main characteristics (i.e., its IP address, its corresponding zone, and region). Hereafter, we denote the VM hosting a component under test by VM under test (VMUT) and the VM hosting the test component by test VM (T_VM).

26.6 Related Work

In this section we give an overview of contributions from the literature and from our previous work related to the security of IoT applications in smart cities.

Although the security protocols are well elaborated in the Internet domain, it is still not fully clear how these existing IP security protocols and architectures can be adapted and deployed in the context of distributed and heterogeneous environment like the Internet of Things (IoT). From its appearance as a promiscuous technology, several works are addressing the security problems of the IoT [4, 8, 12, 13, 26, 28, 32, 36], but until now there is no sufficient solutions that meet the users' requirements and performance needs. In the following, we will provide an overview of the most important related works and clarify the contributions of our proposal in comparison to the state of the art.

The authors in [4] presented the security contributions of the SMARTIE work, which aims to provide secure IoT data management for smart cities. The authors first classified attacks to internal and external. Internal attacks are caused by devices and/or users of the smart city environment. Internal attackers are more dangerous than external ones as they have detailed knowledge about the infrastructure, they have access to part of the systems and they hold some keys. Internal attackers can be users or administrators of the smart city systems or any hackers that succeed to compromise a component of the system. Note that, due to the diversity of the IoT devices and their spread in different locations, device compromise attack is easier in IoT than in classical networks. On the other hand, external attackers may try to access private data from users, components, or subsystems of the IoT environment. Moreover, as in IoT some components are controlled remotely, attackers can exploit these features to gain control and manipulate victims devices. Two main security mechanisms are defined in [4] to address the above issues. First, DCapBAC [13] is an authorization scheme that takes access control decisions before the actual service is accessed. It does this by giving a signed authorization token to a user who is asking for any particular service or functionality offered by a thing. The authorization token is sent along with a request to the thing that verifies the validity of the request and the authorization token, delivering the requested data, if successful. Second, PrivLoc [33] offers secure location-based services, in particular a secure geo-fencing service that alerts users if objects enter or leave a defined area. Location-based services are increasingly gaining importance. Not only end

users but also companies can make use of location data to track assets (e.g., public transport services, users looking for transportation, or logistics companies). PrivLoc scrambles location information in a way that allows computation on intersections of scrambled geometric objects, which is the main operation behind a geo-fencing service.

In paper [32], the authors proposed OSCAR (object security architecture for the Internet of Things), an architecture for end-to-end security in the Internet of Things. It is based on the concept of object security that relates security with the application payload. The architecture includes authorization servers that provide clients with access secrets that enable them to request resources from constrained nodes. The nodes reply with the requested resources that are signed and encrypted. Although this architecture solves some of disadvantages of Datagram Transport Layer Security (DTLS) since it supports multicast, asynchronous traffic, and caching, it has certain limitations. Indeed, OSCAR is vulnerable to the replay attack and its performance is affected with the eventual usage of larger elliptic curve cryptography (ECC) curves. In [12], an architectural reference model-compliant framework was proposed. This framework emphasizes on security and privacy aspects to be used on smart buildings scenarios. Additionally, authors proposed an extension of the security functional components of the reference architecture in order to enable more flexible sharing models, in which the physical context information is considered as a first-class component in order to realize the so-called context-aware security on IoT scenarios. As an instantiation of this framework, a platform for services management on smart buildings has been deployed and extended to offer both user-centric services, like comfort and energy saving, and discovery and security functionality for such services. The feasibility of the proposed mechanisms has been demonstrated through the instantiation of the platform and its evaluation in a smart building used as reference [4].

From a standardization perspective, the recently standardized constrained application protocol (CoAP) was proposed as a lightweight alternative to the HTTP protocol for web-based IoT applications, but security does not keep up. For this purpose, the IETF has thus taken a position to reuse the datagram transport layer security (DTLS), the all-round point-to-point security protocol, to secure the communication channel between a constrained device running CoAP and a client [32]. However, apart from its current incompatibility with caching and multicast traffic, the DTLS approach has an important impact on scalability: Memory limitations of constrained nodes restrict the number of DTLS sessions. In IoT scenarios such as smart cities in which a large number of clients may communicate with constrained CoAP nodes, the limitations lead to a considerable load that translates to an increased energy consumption and a shortened lifetime [32]. Several works have thus been proposed to overcome these limitations of DTLS.

The DTLS in constrained environments (DICE), an IETF working group, was formed to add multicast security to DTLS [15]. In [15], the authors present a method for securing IPv6 multicast communication based on the DTLS which is already supported for unicast communication for CoAP devices. They deal with the adaptation of the DTLS record layer to protect multicast group communication,

assuming that all group members already have the group security association parameters in their possession. The adapted DTLS record layer provides message confidentiality, integrity, and replay protection to group messages using the group keying material before sending the message via IPv6 multicast to the group [15]. However, the authors did not present how group members can agree on the group security association.

In [33], the authors presented Lithe—an integration of DTLS and CoAP for the IoT. Lithe proposes a novel DTLS header compression scheme that aims to reduce the energy consumption by leveraging the 6LoWPAN standard based on reducing the number of transmitted bytes while maintaining DTLS standard compliance.

Granjal et al. [9], described mechanisms to enable security at the network layer, based on the IPSec protocol, and at the application layer, based on the DTLS protocol, and performed an extensive experimental evaluation study with the goal of identifying the most appropriate secure communication mechanisms and the limitations of current sensing platforms for supporting end-to-end secure communications in the context of Internet-integrated sensing applications [9]. These results showed a similar performance of the two approaches, except in the case when DTLS is additionally used to exchange keys with the elliptic curve Diffie-Hellman exchange.

Heer et al. [10] discussed the applicability and limitations of existing Internet protocols and security architectures in the context of IoT. They presented challenges and requirements for IP-based security solutions and highlighted specific technical limitations of standard IP security protocols. It was indicated that for supporting secure IoT, its security architecture should fit the life cycle of a thing and its capabilities, and scale from small-scale ad-hoc security domains of things to large-scale deployments, potentially spanning several security domains. Security protocols should further take into account the resource-constrained nature of things and heterogeneous communication models. Lightweight security mechanisms and group security that are feasible to be run on small things and in IoT context should be developed, with particular focus on possible DoS/DDoS attacks. In addition, cross layer concepts should be considered for an IoT-driven redesign of Internet security protocols.

The authors in [35] addressed the routing protocol for low-power and lossy networks (RPL) attacks and they provided a comprehensive analysis of IoT technologies and their new security capabilities that can be exploited by attackers or IDSs. One of the major contributions in [35] is the implementation and demonstration of well-known routing attacks against 6LoWPAN networks running RPL as a routing protocol. The implemented attacks are selective-forwarding attacks (where malicious nodes selectively forward packets and therefore can achieve a DoS attack), sinkhole attacks (where a malicious node advertises an artificial beneficial routing path and attracts many nearby nodes to route traffic through it), HELLO flood attacks (where the attackers by broadcasting a HELLO message with strong signal power and a favorable routing metric can introduce himself as a neighbor to many nodes, possibly the entire network), wormhole attacks, clone ID, and

Sybil attacks. In order to mitigate these attacks, the authors proposed an intrusion detection system (IDS), called SVELTE [27].

SVELTE [27], an intrusion detection system for the IoT was designed, implemented, and evaluated against routing attacks such as spoofed or altered information, sinkhole, and selective-forwarding. SVELTE's overhead is small enough to deploy it on constrained IoT nodes with limited energy and memory capacity. However, SVELTE assumes that it has access to the border router of the network to place heavyweight IDS parts there. This assumption is not always possible. For example in a smart city application, the messages can be routed over a cellular station that belongs to the network of another owner and therefore we did not have access to the border router.

The previously cited proposed solutions can be classified to three categories which are application layer security solutions [15, 33], network layer security solutions [27, 35], and context-aware security solutions [4, 12, 32]. Context-aware security solutions are very dependent to the specific characteristics of the applications use case and so suffer from the lack of inter-operability. Moreover, the existing solutions have a high computation and communication cost that make them inadequate to resource-constrained things. Network security solutions are limited to attacks related to network layer and so cannot mitigate attacks that target the application layer and cannot provide some security services such as authentication and access control.

Contrary to [27, 35] that proposed an IDS that is limited to routing attacks. In this work we aim to extend the functionality of the IDS and to address also the application layer attacks that target the CoAP protocol. This kind of IDS will present a first line of defense and will mitigate several attacks such as the DoS attack. In addition, this work will focus on providing security based on the DTLS protocol as there are several attempts to make this protocol as the standard for security in the IoT. Therefore, we will propose enhancements to the DTLS protocol to fit the IoT objects. Moreover, we will focus on not resolved aspects such as group key management and multicast communication.

The authors of [6] propose a good survey on more than one hundred publications on model-based security testing extracted from the most relevant digital libraries and classified according to specific criteria. Even though this survey reports on a large number of articles about MBST it does not contain any reference to IoT applications or smart cities. Contrary to that the authors of [1] propose a model-based approach to test IoT platforms (with tests provided as services) but they do not deal with security aspects at all.

26.7 Conclusion

In this work we aimed to combine these two directions, namely: model-based testing and security testing for IoT applications in smart cities. For that purpose we took advantage of our previous findings [5, 16, 20, 22] related to these fields. Moreover,

we extended the notions proposed in the survey [5] to the case of IoT applications. We also exploited our previous results about test techniques of dynamic distributed systems [16, 20].

Our work is at its beginning and a lot of efforts are needed at all levels on both theoretical and experimental aspects. First we need to deal with modeling issues. In this respect we need to extend our modeling formalism and to identify the particular elements of IoT applications to model (using extended timed automata). Models must not be big in order to avoid test number explosion. For that purpose we need to keep an acceptable level of abstraction. As a second step we have to adapt our test generation and selection algorithms to take into account security requirements of the applications under test. The new algorithms must be validated theoretically and proved to be correct. In the same manner we need to upgrade our tools to implement new obtained algorithms. We also need to validate our approach with concrete examples with realistic size. Finally we propose to adopt the same methodology as in [22] to combine security and load tests for IoT applications.

References

1. Ahmad, A., Bouquet, F., Fourneret, E., Le Gall, F., Legeard, B.: Model-based testing as a service for IOT platforms. In: Margaria, T., Steffen, B. (eds.) Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications, pp. 727–742. Springer, Cham (2016)
2. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. **126**, 183–235 (1994)
3. Behrmann, G., David, A. and Larsen, K.G.: A tutorial on uppaal. In: Bernardo, M., Corradini, F. (eds.)International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004. Revised Lectures, vol. 3185, LNCS, pp. 200–237. Springer, Berlin (2004)
4. Bohli, J.-M., Skarmeta, A., Moreno, M.V., García, D., Langendörfer, P.: Smartie project: secure IoT data management for smart cities. In: 2015 International Conference on Recent Advances in Internet of Things (RIoT), vol. 00, pp. 1–6 (2015)
5. Cheikhrouhou, O.: Secure group communication in wireless sensor networks: a survey. J. Netw. Comput. Appl. **61**, 115–132 (2016)
6. Felderer, M., Zech, P., Breu, R., Büchler, M., Pretschner, A.: Model-based security testing: a taxonomy and systematic classification. Softw. Test. Verif. Reliab. **26**(2), 119–148 (2016)
7. Gao, J., Bai, X., Tsai, W.-T.: Cloud testing- issues, challenges, needs and practice. Softw. Eng. Int. J. **1**(1), 9–23 (2011)
8. Garcia-Morchon, O., Kumar, S., Keoh, S.L., Hummen, R., Struik, R.: Security Considerations in the IP-Based Internet of Things, Internet-Draft draft-garcia-core-security-06, Internet Engineering Task Force, Fremont (2013). Work in Progress
9. Granjal, J., Monteiro, E., Sá Silva, J.: On the effectiveness of end-to-end security for internet-integrated sensing applications. In: 2012 IEEE International Conference on Green Computing and Communications, pp. 87–93 (2012)
10. Heer, T., Garcia-Morchon, O., Hummen, R., Keoh, S.L., Kumar, S.S., Wehrle, K.: Security challenges in the ip-based internet of things. Wirel. Pers. Commun. **61**(3), 527–542 (2011)
11. Henzinger, T.A., Manna, Z., Pnueli, A.: What good are digital clocks? In: Kuich, W. (ed.) Automata, Languages and Programming, pp. 545–558. Springer, Berlin (1992)
12. Hernández-Ramos, J.L., Moreno, M.V., Bernabé, J.B., Carrillo, D.G., Skarmeta, A.F.: SAFIR: secure access framework for IoT-enabled services on smart buildings. J. Comput. Syst. Sci. **81**(8), 1452–1463 (2015)

13. Hernández-Ramos, J.L., Jara, A.J., Marin, L., Gómez, A.F.S.: Dcapbac: embedding authorization logic into smart things through ECC optimizations. Int. J. Comput. Math. **93**(2), 345–366 (2016)
14. Hessel, A., Larsen, K.G., Nielsen, B., Pettersson, P., Skou, A.: Time-optimal real-time test case generation using uppaal. In: Petrenko A., Ulrich, A. (eds.) Formal Approaches to Software Testing, pp. 114–130. Springer, Berlin (2004)
15. Keoh, S., Kumar, S., Garcia-Morchon, O., Dijk, E., Rahman, A.: DTLS-Based Multicast Security for Low-Power and Lossy Networks (LLNs). Internet-Draft Draft-keoh-dice-multicast-security-08, Internet Engineering Task Force, Fremont (2014). Work in Progress.
16. Krichen, M.: A formal framework for black-box conformance testing of distributed real-time systems. IJCCBS **3**(1/2), 26–43 (2012)
17. Krichen, M., Tripakis, S.: Black-box conformance testing for real-time systems. In: Graf, S., Mounier, L. (eds.) Model Checking Software, pp. 109–126. Springer, Berlin (2004)
18. Krichen, M., Tripakis, S.: Conformance testing for real-time systems. Form. Methods Syst. Des. **34**(3), 238–304 (2009)
19. Krichen, M., Cheikhrouhou, O., Lahami, M., Alroobaea, R., Jmal Maâlej, A.: Towards a model-based testing framework for the security of internet of things for smart city applications. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 360–365. Springer, Cham (2018)
20. Lahami, M, Krichen, M., Jmaïel, M.: Safe and efficient runtime testing framework applied in dynamic and distributed systems. Sci. Comput. Program. **122**(C), 1–28 (2016)
21. Lee, D., Yannakakis, M.: Principles and methods of testing finite state machines-a survey. Proceedings of the IEEE **84**(8), 1090–1123 (1996)
22. Maâlej, A.J., Krichen, M.: A model based approach to combine load and functional tests for service oriented architectures. In: Proceedings of the 10th Workshop on Verification and Evaluation of Computer and Communication System, VECoS 2016, Tunis, October 6–7, 2016, pp. 123–140 (2016)
23. Mell, P., Grance, T.: The Nist Definition of Cloud Computing (2011)
24. Myers, G.J., Sandler, C.: The Art of Software Testing, Wiley, Hoboken (2004)
25. Nguyen, K.T., Laurent, M., Oualha, N.: Survey on secure communication protocols for the internet of things. Ad Hoc Netw. **32**, 17–31 (2015)
26. Park, S.D., Kim, K.-H., Haddad, W., Chakrabarti, S., Laganier, J.: IPv6 over Low Power WPAN Security Analysis. Internet-Draft draft-daniel-6lowpan-security-analysis-05, Internet Engineering Task Force, Fremont (2011). Work in Progress
27. Raza, S., Wallgren, L., Voigt, T.: Svelte: real-time intrusion detection in the internet of things. Ad Hoc Netw. **11**(8), 2661–2674 (2013)
28. Roman, R., Zhou, J., Lopez, J.: On the features and challenges of security and privacy in distributed internet of things. Comput. Netw. **57**(10), 2266–2279 (2013)
29. Springintveld, J., Vaandrager, F., D'Argenio, P.R.: Testing timed automata. Theor. Comput. Sci. **254**(1), 225–257 (2001)
30. Tretmans, J.: Testing concurrent systems: a formal approach. In: Baeten, J.C.M., Mauw, S. (eds.) CONCUR'99 Concurrency Theory, pp. 46–65. Springer, Berlin (1999)
31. Tripakis, S.: Fault diagnosis for timed automata. In: Damm, W., Olderog, E.R. (eds.) Formal Techniques in Real-Time and Fault-Tolerant Systems, pp. 205–221. Springer, Berlin (2002)
32. Vucinic, M., Tourancheau, B., Rousseau, F., Duda, A., Damon, L., Guizzetti, R.: OSCAR: object security architecture for the internet of things. CoRR, abs/1404.7799 (2014)
33. Vučinić, M., Tourancheau, B., Rousseau, F., Duda, A., Damon, L., Guizzetti, R.: Oscar. Ad Hoc Netw. **32**(C), 3–16 (2015)
34. Walewski, J.: Internet-of-Things Architecture IOTA Project Deliverable d1.2 - Initial Architectural Reference Model for IOT (2018)
35. Wallgren, L., Raza, S., Voigt, R.: Routing attacks and countermeasures in the RPL-based internet of things. Int. J. Distrib. Sens. Netw. **9**(8), 794326 (2013)
36. Yan, Z., Zhang, P., Vasilakos, A.V.: A survey on trust management for internet of things. J. Netw. Comput. Appl. **42**, 120–134 (2014)

Index

R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2