Journal of Communications and Information Networks, Vol.2, No.3, Sept. 2017
DOI: 10.1007/s41650-017-0031-9

Research paper

Special Focus on Future Internet

Big data storage and management in SaaS applications

Xi Zheng[1*], Min Fu[1], Mohit Chugh[2]

1. Macquarie University, Macquarie Park NSW 2109, Australia
2. Deakin University, Geelong VIC 3220, Australia

* Corresponding author, Email: jameszhengxi@utexas.edu

Abstract: As an important service model for advanced computing, SaaS uses a defined protocol that manages services and applications. The popularity of advanced computing has reached a level that has led to the generation of large data sets, which is also called Big data. Big data is evolving with great velocity, large volumes, and great diversity. Such an amplification of data has brought into question the existing database tools in terms of their capabilities. Previously, storage and processing of data were simple tasks; however, it is now one of the biggest challenges in the industry. Experts are paying close attention to big data. Designing a system capable of storing and analyzing such data in order to extract meaningful information for decision-making is a priority. The Apache Hadoop, Spark, and NoSQL databases are some of the core technologies that are being used to solve these issues. This paper contributes to the solutions to the issues of big data storage and processing. It presents an analysis of the current technologies in the industry that could be useful in this context. Efforts have been focused on implementing a novel Trinity model, which is built using the lambda architecture with the following technologies: Hadoop, Spark, Kafka, and MongoDB.

Keywords: SaaS, big data, Apache Hadoop, Apache Kafka, lambda architecture, NoSQL

Citation: X. Zheng, M. Fu, M. Chugh. Big data storage and management in SaaS applications [J]. Journal of communications and information networks, 2017, 2(3): 18-29.

1 Introduction

Owing to the popularity of advanced computing, big data is growing. Big data refers to a very high volume of data that cannot be handled using traditional database methods. Traditional database practices have reached their limits in terms of performance and are not capable of processing such data with good results. Several new storage techniques have emerged with the increase in data volumes. The majority of the database practitioners have only emphasized on structured data, but the standard expansion of such huge data comprises 80% of unstructured and semi-structured data[1] including pictures, videos, and texts. Big data has influenced several real-world fields such as banking, medication, and education. The three vital elements to be considered while defining big data are volume, variety, and velocity[2,3]. The volume of big data will be enormous. The scalability of the databases and their capability of processing such big data would be a prime requirement for any application, including data-driven cloud applications, large-scale mobile cooperative web applications, and user-oriented big-data-driven systems. In particular, multi-copy data dissemination and delay-constrained data query in large-scale mobile networks have become the focus of many studies[4,5]. Furthermore, the storage,

Manuscript received Jun. 18, 2017; accepted Aug. 28, 2017

security, and processing of big data are major concerns; this study addresses these challenges. Academic and industrial experts are paying close attention to these issues. Currently, with the speed at which big data is growing, its size will soon reach the zettabyte[2]. New technologies and techniques are emerging: Apache Spark, Hadoop, Apache Storm, and NoSQL databases. There are some disadvantages of using RDBMSs (Relational Database Management Systems). The mass generation of data in the big data era would require a relational database for storing millions of records and processing them in real time. However, the RDBMS is not sufficiently efficient[6-8]. In addition to this, the RDBMS is also not suited to working in a distributed manner. This has led to the emergence of NoSQL databases that provide better data accessibility for unstructured or semi-structured data. There are multiple NoSQL databases, and we analyze the MongoDB and Cassandra databases in this study. Cost is also an aspect that drives our research[9]. Big data is obtained on using digitalized media or any data-intensive technologies. It is impossible to process such huge data using traditional database technologies. This is because the relational databases were developed at a time when different software and hardware specifications were used and to be associated often with structured data[8]. In this study, we provide a deep analysis of current technologies and select the best one. We have envisioned a new architecture, the Trinity model, for solving the storage and processing issues of big data. It is not only more practical but also more affordable. It is practical because the Trinity model delivery is based on certain open-source middleware; it is affordable because the data storage does not incur any monetary cost. Our proposed model takes into consideration two important aspects: scalability and processing performance. We demonstrate the validity of our solution through our evaluation. The contributions of this paper are as follows: 1) We propose a novel big data processing solution, called the Trinity model. To the best of our knowledge, such a model has not been previously proposed. 2) We conduct a systematic analysis of the existing big data processing solutions in order to provide a better understanding of the state-of-the-art big data technology. 3) We present a complete justification of our proposed solution and demonstrate its utility through an evaluation. The remainder of this paper is organized as follows: section 2 presents the background and motivation for this study; section 3 discusses various big data processing platforms; section 4 illustrates our proposed solution; section 5 presents the evaluation of the model; section 6 presents the related works; and section 7 presents the conclusions of this study and our future work.

2 Background and motivation

The big data technologies are first categorized as database solutions and processing solutions that are available in the market: MongoDB, Cassandra, Apache Hadoop, Apache Spark, and Apache Kafka. NoSQL databases are undoubtedly the future in terms of improving data accessibility for unstructured data. They are categorized as document-based, key-value-based, graph-based, and column-based[10,11]. They are becoming the preferred choice in the industry for storing big data[11]. They are designed while keeping in mind the best possible method of scalability for unstructured data. This study analyzes such NoSQL databases with their pros and cons. The MongoDB, as a document database, is selected for the Trinity model for improved storage. The advantageous features of the MongoDB are the use of a schema-less data model, elastic scalability, and data replication[12,13]. Apache Hadoop is the defacto standard that is being used by the majority of the industry as a base for any big data application[14,17]. It is one of the open-source tools available to the community for in-depth research. We found that Apache Hadoop still has flaws that will be discussed in later sections, which can be overcome by using external fundamentals. The analysis of such big technologies provided us with an understanding of the success of hybrid solutions, as no single technology is capable of dealing

with such big data alone[6,9,14]. Attempts at combining Hadoop and several other technologies (such as Apache Kafka) lead us to this successful novel project. The design of the Trinity model is based on the lambda architecture that provides a platform for processing both batch and real-time data simultaneously. This study claims that, if the Trinity model is implemented, it would be capable of solving the major issues of big data. Apache Kafka[15] on Hadoop, another open-source tool for distributed data, exhibits a highly desirable performance. It has a promising future in real-time applications. It helps our model in the queuing of the incoming data into the memory, which can further be used by Spark for better results. The lambda architecture[8,16] is being used to incorporate the suitable technologies. There exists extensive debate regarding the incorporation of the right technology on the lambda architecture for improved workflow. Our literature review and in-depth analysis allow us to choose the best technology for batch processing (MapReduce) and real-time processing (Apache Spark). Storage is always a requirement of any big data application[11,14]. The application should have scalable data storage that allows it to save as much data of all types as it requires.

3 Analysis of big data platforms

In this section, we conduct an in-depth analysis of existing big data systems and platforms.

3.1 Hadoop

Apache Hadoop is a Java-based programming framework that is used for processing huge sets of data[14]. The advantage of Hadoop is that it processes data that is placed in a distributed computer environment. The Hadoop architecture is used by several large companies such as Facebook, Google, Yahoo, and IBM[6,14]. Fig. 1 shows the two core components of Hadoop[17]. Hadoop has its own file system, i.e., the HDFS (Hadoop Distributed File System), which facilitates fast data transfers and prevents system failure with the help of its core function called MapReduce. The MapReduce algorithm breaks the big data into small chunks, distributes it on multiple servers/nodes, and then performs operations[17,18].

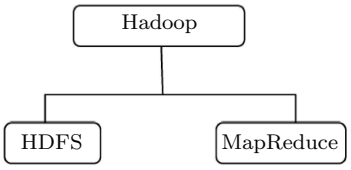

Figure 1 Components of Hadoop

The HDFS is considered as highly fault-tolerant and is designed for special deployment in minimum-cost hardware[17,18]. It has similarities with other currently used file systems but its unique features are sufficiently significant to make it a giant architecture in today's industry. Fig. 2 depicts the architecture of the HDFS. The HDFS stores file system meta data and application data separately[17]. The HDFS uses the master/slave scheme. The Hadoop cluster is divided into the NameNode and DataNodes. The NameNode is treated as a master node, the function of which is to manage the namespaces and accesses on file. There will always be only one NameNode but there could be innumerable DataNodes, at least one per node in a given cluster. DataNodes are treated as slaves in the architecture that contains the actual application data. However, a file is internally divided into numerous blocks that are stored in a DataNode[18].

The operations handled by the NameNode include opening, closing, and renaming files or directories[18]. DataNodes are given the responsibility of handling read and write operations. They are also allowed to manage replication[18].

MapReduce is a core functionality of Hadoop. It was first introduced by Google in 2004[18] with the sole objective of supporting the distributed computing of large data. It is one of the popular programming models for processing large sets of data located on several servers[6,17]. The users are required to specify a map function that processes a key/value pair in order to obtain another intermedi-

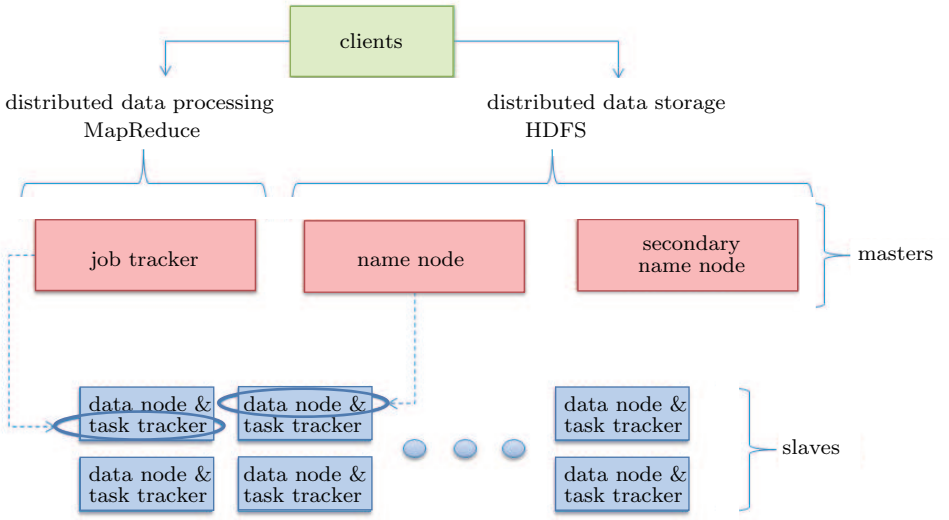

Figure 2 Architecture of Hadoop distributed file system

ate key/value pair. Reduce functions are required to be defined as well so that it merges all intermediate values for a single key in the end[18].

3.2 Spark

In 2009, a team of a few researchers and developers of the University of California, Berkley initiated a project for designing a unified engine for processing distributed data[19]. Spark is a processing model that is similar to MapReduce but has some major features for processing real-time data[19]. The function of Spark extends over that of MapReduce in terms of one of its major functionalities: RDDs (Resilient Distributed Datasets), which is the primary data abstraction in Apache Spark[19,20]. Through the use of RDDs, Spark is capable of capturing a broader workload of processing tools that includes SQL support, streaming, machine learning, and graph processing[21]. Spark has some important features[19]: 1) it would be easy to develop applications using Spark because of its unified API. 2) Spark has the combined support of machine learning, graph processing, and streaming, which was not possible with previous tools. 3) It processes data in its memory, which makes its processing performance superior to that of other processing models.

Spark streaming[22] is an extension of Spark that provides high scalability, throughput, and fault-tolerant streaming of real-time data. Data can have inputs from multiple sources such as Apache Kafka, TCP sockets, or Apache Flume. The input is analyzed well and processed via complex algorithms that contain high-level functions. These high-level functions involve mapping, reducing, joining, and windowing[22]. However, once the data is processed, it is scattered well in a file system or database. The streaming requires a great amount of time for processing[23]. However, Spark saves the currently used data into HDFS and then builds new results by working on these saved historical data. The integration of powerful tools for addressing big data problems with high performance and which have easy access to unified programming APIs (including Python, Scala, and Java) is what has made Spark one of the major projects in Apache[19-24]. In order to provide greater ease of use, the Apache Spark has released its own SQL-like programming API for the users who feel uncomfortable with other languages[23].

3.3 Apache Storm

Apache Storm is considered as another real-time data processing system that is fault-tolerant and has a distributed architecture[25]. It uses directed acyclic

graphs, which are also called topologies, for its core workflow. The process of abstraction of data is called a Stream, which has a continuous and sequencing working nature with tuples[25]. Streaming is carried out by Spouts, which gets data into the Storm ecosystem from external sources. Then, there are entities responsible for transforming the incoming data streams, called Bolts. Bolts consist of MapReduce, filtering, and aggregation-like functionalities for dealing with data. Bolts is also designed to communicate with external databases if necessary[25].

Fig. 3 shows Strom Trident. This is a process of abstraction inside Storm that is responsible for performing micro-batching. This also facilitates high-level operations such as group by, aggregate, and count. The micro-batching in Storm is achieved by breaking up the incoming data into smaller batches of tuples[25]. Some important features are as follows[19]: 1) The applications would be easy to develop with Spark because of its unified API. 2) Spark has the combined support of machine learning, graph processing, and streaming, which was not possible with previous tools. 3) It processes data in its memory, which makes its processing performance superior to that of other processing models.

3.4 Lambda architecture

Apache Hadoop and MapReduce are widely used for storing and processing big data. Hadoop is the defacto ecosystem for designing any architecture in terms of big data over multiple servers. Hadoop is very popular for handling the volume trait[16]. MapReduce focuses on batch processing and aims at processing long running background processes[14,26]. Therefore, we still require better techniques for handling the velocity of acting on continuous recent data. The requirement for processing batch and real-time data in parallel and adding the real-time computation to the batch-processing systems resulted in a framework called the lambda architecture. The main objective of the lambda architecture is to provide a generic approach for processing big data with high scalability and fault tolerance[16,26]. He developed this idea based on his experience in dealing with distributed data processing systems in the case of Twitter. The lambda architecture has three layers: the batch layer, speed layer, and serving layer.

3.5 NoSQL databases

NoSQL databases are the preferred choice among experts and developers in the industry nowadays[27]. The major issue for them is that the rapidly growing data requires an appropriate design for storage in terms of efficient data analysis[10,27]. Owing to such requirements, the traditional databases became obsolete, and NoSQLs were taken into consideration while keeping properties such as a schema-less data model, elasticity, and scalability in mind[12]. Relational databases were designed in a different era for different software and hardware requirements[6]. They were not designed for handling extra scalability. Most of the data is noisy or unstructured and cannot be handled by RDBMSs. Therefore, these systems are not able to cope with such challenges. The NoSQLs are popular in maintaining consistent models in contrast to the RDBMS in the case of operations such as retrieval and collection of data[8]. As previously mentioned, the NoSQL is the preferred choice when the storing and processing of large data is a prime requirement. The simplicity of the data model design, horizontal scalability, and prior availability are some of the important features of NoSQL. Indrawan-Santiago mentioned in Ref. [6] the use of NoSQLs as a compliment to RDBMSs, which resulted in the improvement of the organization's data management capabilities. In addition to this, the authors in Ref. [7] compared the NoSQL database with the SQL database and found that owing to the large number of records in SQL, the inserting time is one of the major problems that must be solved. NoSQL databases are required in the application areas of e-commerce, web application, location-based services, social media, Internet-of-Things data storage, etc. These areas require predictive analytics and real-time data processing and are characterized by huge data sets from various sources. NoSQL

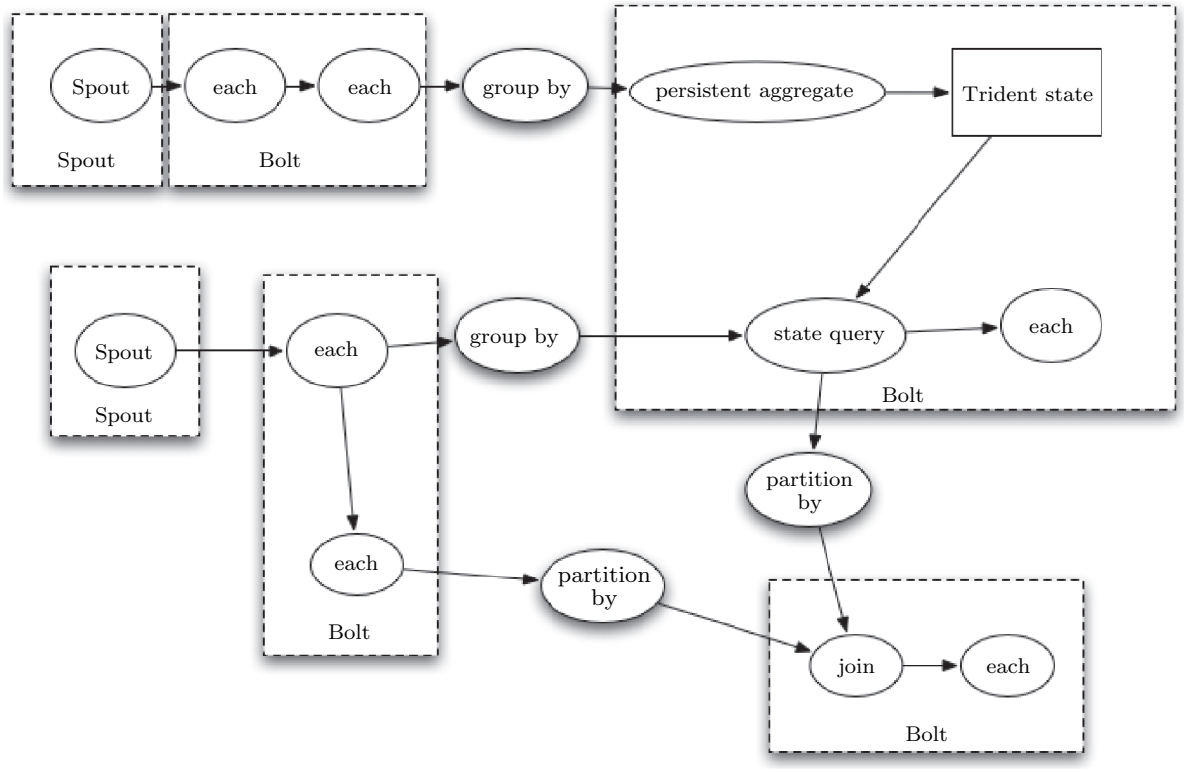

Figure 3 Architecture of Apache Storm

is capable of meeting these requirements[7]. The NoSQL database is divided into four major categories: column-based, document-based, key-value-based, and graph-based[7]. The best practices involve the selection of the best NoSQL database as per their application requirements depending on the data model, performance (read and write), runtime, and throughput[7,10,27].

4 Our solution: Trinity model

The Trinity model is designed to overcome the issue of storing and processing big data. Its structure is illustrated in Fig. 4. It consists of three components: the data model, RDBMS, and NoSQL. These three components cooperate with each other and work as an ecosystem. The structure and characteristics of the data that are formulated in the data model determine whether the data should be stored and processed in the RDBMS or in NoSQL. The RDBMS is more suitable for storing and processing structured data, while NoSQL is more convenient for storing and processing unstructured or semi-structured data.

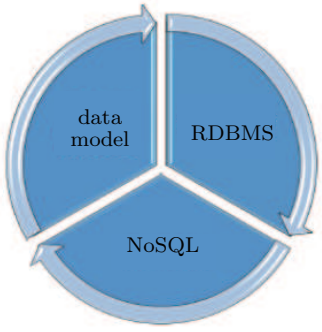

Figure 4 Trinity model

The data model is a core pillar that identifies the type of data to be stored by the application. This would also indicate how the data would be transferred between the databases for operations such as retrieval. Once the data passes this layer and the decision is made, the storage and processing will depend on the database handling rules. This process is shown in Fig. 5.

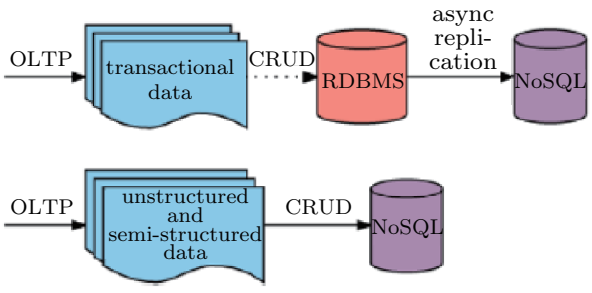

Figure 5 Trinity model processing architecture

The data model in our architecture has resolved the issue of handling data efficiently. There exists a problem of storing large sets of data in the case of big data. The majority if the data is in the form of semi-structured or unstructured data instead of structured data. The data that can be organized and easily searched is referred to as structured data. It is stored and organized at a high level. In contrast, semi-structured and unstructured data do not follow any defined data model and are difficult to operate with. The first operation performed using the Trinity model is the organization of such data before it is stored or processed. However, the data is meant to reside either in the RDBMS or NoSQL. The relational database is responsible for handling transactional data and can only store structured data. In addition to this, NoSQL is responsible for storing semi-structured or unstructured data. Relational databases are well designed for handling OLTP (Online Transaction Processing). This is because of the properties of the RDBMS, i.e., ACID (Atomicity, Consistency, Isolation, Durability) properties, which are a few of the pre-requisites for OLTP. Once the data is stored in the RDBMS; it also gets replicated in NoSQL. This provides high availability to the system. The unstructured or semi-structured data would directly surpass this layer of the Trinity model and reside in the NoSQL database. NoSQL would also look after the OLAP (Online Analytical Processing). These systems are designed to facilitate decision making. These decisions form the basis for making operational decisions. Manipulation of the data allows the analysis of the measurement hierarchy and execution of operations such as classifications and predictions. NoSQL databases are beneficial owing to their ability to manage huge data[6] and flexibility over relational databases. There have been multiple instances in which experts have questioned the lack of data modeling for OLAP in RDBMSs and have suggested migrating it to NoSQL databases. Efforts have also been made to explain the proposed model. The proposed model is one of the major promising frameworks that aim for efficient storage and faster processing for the SaaS (Software as a Service) applications that involve the handling of big data. Fig. 6 shows a block diagram of our proposed model.

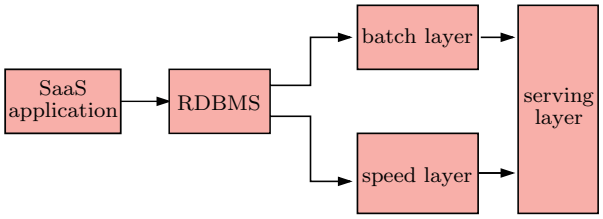

Figure 6 Block diagram of proposed model

The block diagram clearly shows the movement of data in the simplest manner. The data is collected via a SaaS application interface and is transferred to the RDBMS. The data is analyzed by the data model that is imposed as the core functionality of our model, which will determine whether the data is required to be stored in the RDBMS. If the data is structured, it will directly reside in the RDBMS; if the data is unstructured or semi-structured, it will directly be transferred to the NoSQL database after passing through the batch layer. Such a separation increases the data-processing efficiency. This architecture also includes the three layers of the lambda architecture. A complete description of the proposed model is shown in Fig. 7.

The diagram shown above clearly shows some major components of the architecture: 1) RDBMS; 2) Apache Kafka; 3) batch layer; 4) speed layer; and 5) serving layer. The SaaS application interface is at the start of the workflow. The application has the ability to collect and transfer data to the internal architecture. The function of data modeling lies ahead of this interface and is given the responsibility

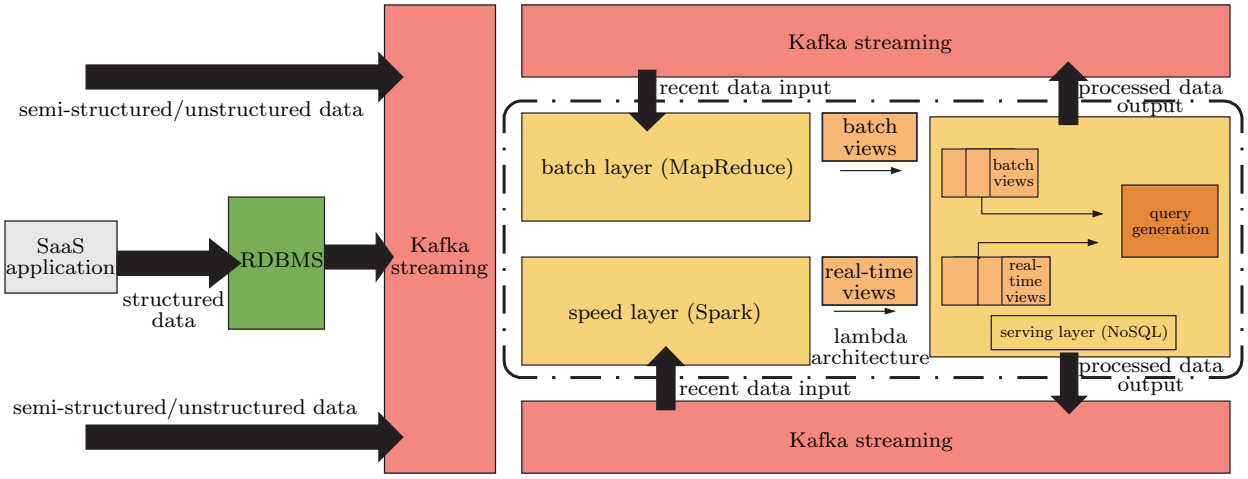

Figure 7 Proposed model for storage and processing of big data

of determining whether the data is structured or unstructured. If the data modeling is unavailable, the analytics would not work. Apache Kafka is taken into consideration and any process can put data on it, can store data into it, and it will be used as a data transportation pipeline for all three lambda layers. Kafka is also used to publish both models and model updates for consumption by the speed and serving layers. The core functionality of Kafka leads this architecture in terms of activity tracking, queuing the data stream in a message-like fashion, fast and live streaming, and log aggregation. Kafka provides the system with high reliability. It replicates the incoming data and its strength lies in supporting and distributing for the batch, speed, and serving layers. In the case of any failure, it also looks after the restoration of the lost data. Kafka maintains a log table of the data streaming. Kafka also enhances the performance of the proposed model. It delivers a high rate of throughput for both the SaaS API and RDBMS (acting as a producer) and the lambda layers (subscribers). The RDBMS is included in order to look after the OLTP. The ACID properties granted by the relational databases form the platform for our architecture. The batch layer is designed while keeping in mind the offline processing of big data. This is implemented with Hadoop and its MapReduce functionality. It has been configured in a manner that facilitates the reading of the data from Kafka, which acts as a data pipeline for the entire architecture. Thereafter, the architecture is also responsible for processing data in several iteration methods. This would certainly provide accurate and well-managed data. The input of the batch layer can be either structured, semi-structured, or unstructured. The structured data would also be handled by the RDBMS for transactional operations and must be stored in NoSQL. However, the data must also be passed through the batch layer. The output of the batch layer would be the batch views. The speed layer is well implemented in Spark. Spark streaming is the extension of Spark that led the position of best player for processing the real data stream. However, Spark streaming is implemented for processing the streaming of the incoming real data. This real data stream is directed from Kafka. As the process of this layer is aimed at real-time data, the computation is required to be of a short time interval. However, the performance of this layer depends on the speed of the incoming data. The only aim of this layer is to perform the algorithm on the data, irrespective of the degree of accuracy it generates. As the time interval is required to be small, the accuracy of the data can be neglected. The output of this layer is real-time views. The serving layer is the final stage of our proposed model. This layer listens to the queries to be executed in the memory. It processes the batch views and real-time views as per the query being

asked. Several of these types of architectures are deployed for scalability. Hence, the NoSQL databases are the preferred choice of such a layer. These are responsible for writing the results back to Kafka where both of the other layers can again see new data.

5 Evaluation

This section presents the evaluation of the novel architecture in terms of its resilient nature, consistency, and storage structure.

5.1 Resilience

In the case of the correct implementation of an architecture, all possible human errors and hardware failures that would corrupt the data could be neglected. This is because the inclusion of the lambda architecture does not allow updating existing data sets. In addition to this, even if the speed layer (real-time processing layer) fails, there will not be any loss of data. As the write operations are directed and propagated in the batch layer, the updating data would easily be tracked and chased in the case of the failure of the real-time processing layer. Hence, the data in the batch layer would still be accurate, and the results will be synchronized automatically.

5.2 Consistency

The data results processed via the batch layer under the lambda architecture and with the best selected technology will be highly consistent.

5.3 Storage structure

Under the lambda architecture, the data is allowed to be stored under a normalized nature in the batch layer data stores. These data are also meant to be de-normalized according to the requirement of the batch and real-time views. The data required from the application is not directly retrieved from the data stores, instead it is retrieved and fine tuned from all the other layers in the proposed architecture.

6 Related work

A zettabyte of data has been uploaded on the Internet in past year[2]. However, owing to the speed at which the data is being generated, the data is becoming noisier[1-3]. Technologies such as complex event processing and real-time processing are gaining more importance in the industry[2] for the objective of solving big data issues because, as previously mentioned, traditional computation sources lack the desired storage and processing capabilities. The author of Ref. [9] has provided an in-depth analysis of four major research challenges in the field of big data. The aforementioned study also included a conceptual framework requiring at least four categories for all big data applications.

The authors of Refs. [1-3] have attempted to develop a system that can store such big data and analyze such data for decision-making. In terms of architecture, using nanotechnology, they have attempted to design a system that would have a high storage capacity as compared to traditional hard disks and any other extended hard disks[2]. For such a system, a file system is also required. It should thus be noted that there has been a paradigm shift in terms of the migration of the traditional file system to the DFS (Distributed File System). A DFS is a system that allows several candidates to access a file through the same network[2]. In addition to this, an online survey was conducted by the database engine[28] in the context of data management. The DbaaS (Database as a Service) has become popular for cases in which the clients do not want to invest large sums of money to obtain their own data management structure. Instead, they are willing to pay DbaaS providers[10,27]. The DbaaS providers manage the data in their own data centers while allowing the clients to be free of the data management. The chart given below shows the popularity of DbaaS among users for data management. The objective of DbaaS is to charge clients via subscription depending on the data limits the client is willing to use. Some of the big players in the DbaaS market are MS Azure, IBM Cloudant, Google BigTable, and

Amazon Relational Database[29-32]. The authors of Ref. [2] successfully reviewed the current file systems (including the Google file system, IBM general parallel file system, HDFS, and Blobseer & Andrew file system) and found that their scalability, fault tolerance, and availability are still some of the major concerns. A distributed file system is required to have an extendable nature with a low cost[2]. However, using a DFS on multiple OSs (Operating Systems) could be more advantageous. These are also being used by the DbaaS providers for improving the scalability of their databases[2].

The authors of Ref. [33] have also discussed the importance of data deduplication, compression for efficient data storage, and the utilization of bandwidth. Recent studies have shown that experts have attempted to develop deduplication and compression strategies based on the concept of Hashing using MapReduce[10]. He et al. in Ref. [10] made a note regarding several deduplication techniques in which one copy of the duplicate data is maintained and all other nodes are allowed to access that copy. The same paper also discussed a novel method for overcoming the issue of slow performance by traditional storage devices (e.g., hard-disks) using deduplication. Balachandran et al. worked on another technique called compression. He found that if hashes occur more than once in any file, only the first hash value is kept for simplicity[10].

The authors of Ref. [34] have focused their efforts on designing efficient storage techniques for NoSQL databases. Their design comprised several layers inside Hadoop. Data is fetched from the HDFS and stored in flat text files. The deduplication operation is first performed using MapReduce, the output is then stored in a key-value pair, which further creates a pointer table in MongoDB. Compression is then applied in order to realize more efficient storage. Moreover, John Klein and the authors in Ref. [12] compared the performance of MongoDB, Cassandra, and Riak (a key value database). The benchmarking operations used for comparing these three databases are categorized as read-only, write-only, and read-write operations. The results are differentiated depending on the single node configurations. The study reported that the Cassandra performance showed a strong capability of processing 3 200 operations/s, Riak processed 480 operations/s, and MongoDB processed 225 operations/s[12]. The security was and will always be a concern in terms of databases[35,36]. The data pattern is changing rapidly with the rapid development of applications[35,36]. However, there exists a myth that the NoSQL databases are immune to injections[35]. Companies are deploying big data architectures according to their requirements and are very well aware of the security concerns involved. It was stated in Ref. [35] that company repositories are being targeted in order to obtain access to their most valuable information. A US-based retail company faced a loss of $1.1 billion[36], the reason for which was a loophole in their application design that let the attacker perform operations on their database. The loss could have been much higher if this had happened to a financial organization. Although NoSQL databases have been proven to be a compliment to the RDBMS, SQL language should not be used for processing the data. In addition to this, one of the major advantages of these databases is that they are able to change their attributes owing to their weak structure. This also results in the provision of convenient modification while developing interactive applications[35].

7 Conclusion and future work

The era of big data has several more opportunities to offer. The opportunities in the direction of best possible data analytics will result in great benefits in terms of decision-making. The data emerging every second gives rise to some new challenges. The storage and processing of such big data are currently major challenges. There are new techniques and technologies emerging every day that could be used to address these issues. This paper appropriately addresses the aforementioned challenges of big data. This paper presents an in-depth analysis of the major technologies and techniques for the storage and

processing of big data that are currently in use or are new. We discussed the processing tool and data storage process of the various technologies. The processing technologies like Apache Hadoop, Kafka, and Spark are approaching the saturation point of their development as their usage are getting more popular and attracting many more users. Furthermore, data storage has been the key requirement in any application development. The adaption of NoSQL databases has provided an efficient method of storing big data. The MongoDB and Cassandra databases are the most advantageous. This study has also gone the extra mile to propose a novel model for addressing the issue of storing and processing big data depending on the lambda architecture. The lambda architecture has strengthened the model used to process the batch and real-time data simultaneously. The analysis of the technologies allowed us to pick the best one and develop the proposed architecture.

8 Conclusion and future work

Academician Yu made a summary. During the conference, the journal organization works have been well addressed, while the target of the journal in 2016 has been determined. The journal in 2016 will be edited and published quarterly, where 4 issues are defined. A digital periodical platform will be established.

References

[1] Gazal, P. D. Kaur. A survey on big data storage strategies [C]//International Conference on Green Computing and Internet of Things (ICGCIoT), IEEE, 2015: 280-284.

[2] A. Elomari, A. Maizate, L. Hassouni. Data storage in big data context: A survey [C]//International Conference on Systems of Collaboration (SysCo), IEEE, 2016: 1-4.

[3] A. A. Tole. Big data challenges [J]. Database systems journal, 2013, 4(3): 31-40.

[4] Y. Liu, F. Li, Y. Wang. Incentives for delay-constrained data query and feedback in mobile opportunistic crowdsensing [J]. Sensors, 2016, 16(7): 1138.

[5] Y. Liu, A. E. Bashar, F. Li, et al. Multi-copy data dissemination with probabilistic delay constraint in mobile opportunistic device-to-device networks [C]//IEEE 17th International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM), IEEE, 2016: 1-9.

[6] X. B. Chen, S. Wang, Y. Y. Dong, et al. Big data storage architecture design in cloud computing [C]//National Conference on Big Data Technology and Applications, Springer, 2015: 7-14.

[7] P. P. Srivastava, S. Goyal, A. Kumar. Analysis of various nosql database [C]//International Conference on Green Computing and Internet of Things (ICGCIoT), IEEE, 2015: 539-544.

[8] H. L. Zhang, Y. Wang, J. H. Han. Middleware design for integrating relational database and nosql based on data dictionary [C]//International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE), IEEE, 2011: 1469-1472.

[9] R. Ranjan. Streaming big data processing in datacenter clouds [J]. IEEE cloud computing, 2014, 1(1): 78-83.

[10] K. Grolinger, W. A. Higashino, A. Tiwari, et al. Data management in cloud environments: Nosql and newsql data stores [J]. Journal of cloud computing: advances, systems and applications, 2013, 2(1): 22.

[11] Mongodb bringing online big data to business intelligence and analytics [EB/OL]. https://www.mongodb.com/collateral/mongodb-bringing-online-big-data-to-bi-and-analytics, 2017.

[12] H. H. Shahraki, T. J. Gandomani, M. Z. Nafchi. A novel method for evaluation of nosql databases: A case study of cassandra and redis [J]. Journal of theoretical and applied information technology, 2017, 95(6): 1372-1381.

[13] R. Kanwar, P. Trivedi, K. Singh. Nosql, a solution for distributed database management system [J]. International journal of computer applications, 2013, 67(2): 6-9.

[14] P. Córdova. Analysis of real time stream processing systems considering latency [R]. White paper, 2015.

[15] Apache Kafka [EB/OL]. https://kafka.apache.org/intro, 2017.

[16] Lambda architecture [EB/OL]. http://lambda-architecture.net/, 2017.

[17] J. Nandimath, E. Banerjee, A. Patil, et al. Big data analysis using apache hadoop [C]//IEEE 14th International Conference on Information Reuse and Integration (IRI), IEEE, 2013: 700-703.

[18] Apache Hadoop [EB/OL]. http://hadoop.apache.org/, 2017.

[19] M. Zaharia, R. S. Xin, P. Wendell, et al. Apache Spark: a unified engine for big data processing [J]. Communications of the ACM, 2016, 59(11): 56-65.

[20] S. Farook, G. L. Narayana, B. T. Rao. Spark is superior to map reduce over big data [J]. International journal of computer applications, 2016, 133(1): 13-16.

[21] J. Shi, Y. Qiu, U. F. Minhas, et al. Clash of the titans: Mapreduce vs. Spark for large scale data analyt-

ics [C]//Proceedings of the VLDB Endowment, 2015, 8(13): 2110-2121.
[22] Spark streaming [EB/OL]. http://spark.apache.org/streaming/, 2017.
[23] R. S. Xin, J. E. Gonzalez, M. J. Franklin, et al. Graphx: A resilient distributed graph system on Spark [C]//First International Workshop on Graph Data Management Experiences and Systems, ACM, 2013: 2.
[24] S. Gopalani, R. Arora. Comparing Apache Spark and map reduce with performance analysis using k-means [J]. International journal of computer applications, 2015, 113(1): 8-11.
[25] Apache Storm [EB/OL]. http://storm.apache.org/, 2017.
[26] R. Kumar, N. Gupta, S. Charu, et al. Manage big data through NewSQL [C]//National Conference on Innovation in Wireless Communication and Networking Technology, association with the Institution of Engineers (INDIA), 2014.
[27] C. Curino, E. P. C. Jones, R. A. Popa, et al. Relational cloud: a database-as-a-service for the cloud [C]//Fifth Biennial Conference on Innovative Data Systems Research, 2011: 235-240.
[28] DB-Engines [EB/OL]. http://db-engines.com/en, 2017.
[29] Amazon-pricing [EB/OL]. https://aws.amazon.com/ec2/pricing/, 2017.
[30] Azure-pricing [EB/OL]. https://azure.microsoft.com/en-au/pricing/, 2017.
[31] Cloudant-pricing [EB/OL]. https://cloudant.com/product/pricing/, 2017.
[32] Google-pricing [EB/OL]. https://cloud.google.com/bigtable/pricing, 2017.
[33] E. Manogar, S. Abirami. A study on data deduplication techniques for optimized storage [C]//Sixth International Conference on Advanced Computing (ICoAC), IEEE, 2014: 161-166.
[34] V. Bhatia, A. Jangra. Setins: Storage efficiency techniques in no-sql database for cloud based design [C]//International Conference on Advances in Engineering and Technology Research (ICAETR), IEEE, 2014: 1-5.
[35] B. Y. Hou, K. Qian, L. Li, et al. Mongodb nosql injection analysis and detection [C]//IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud), IEEE, 2016: 75-78.
[36] K. Munir. Security model for cloud database as a service (DBaaS) [C]//International Conference on Cloud Technologies and Applications (CloudTech), IEEE, 2015: 1-5.

About the authors

Xi Zheng [corresponding author] is a lecturer/assistant professor in computer science at Macquarie University Australia. He earned a Ph.D. degree in software engineering from the University of Texas at Austin in August 2015. His current research focuses on the design and implementation of middlewares for cyber physical systems (CPS) and Internet of Things in general. His Ph.D. thesis looks into a practical way of bringing formal methods (e.g., temporal logics and automata theories) and physical models (in terms of real time simulation) into CPS runtime verification. (Email: jameszhengxi@utexas.edu)

Min Fu is currently a data mining advisor in Alibaba group. He is also an honorary adjunct fellow in the Department of Computing, Macquarie University, Australia. He received his Ph.D. degree from the University of New South Wales, Sydney Australia. His research interests include: cloud computing, data mining, data analytics, machine learning, cyber security and software architecture. (Email: min.fu@mq.edu.au)

Mohit Chugh is currently a master student in Deakin University. Before pursuing the master's degree, he has worked at EA Games Inc, India for 1 year after gaining bachelor's degree in IT from Sharda University, Greater Noida, India. His research interest lies in database management systems. (Email: mohitc@deakin.edu.au)