# Editors' Introduction and Roadmap

**Sushil K. Prasad, Anshul Gupta, Arnold L. Rosenberg, Alan Sussman, and Charles Weems**

The premise of the NSF-supported Center for Parallel and Distributed Computing Curriculum Development and Educational Resources (CDER) is that every computer science (CS) and computer engineering (CE) undergraduate student should achieve a basic skill level in parallel and distributed computing (PDC). This book is a companion to our 2015 book, the first product of the CDER Book Project.[1] The book series we have embarked on addresses the lack of adequate textbook support for integrating PDC-related topics into undergraduate courses, especially in the early curriculum.[2]

[1]Prasad, Gupta, Rosenberg, Sussman, and Weems. 2015. Topics in Parallel and Distributed Computing: Introducing Concurrency in Undergraduate Courses, 1st Edition, Morgan Kaufmann, ISBN : 9780128038994, Pages: 360. http://grid.cs.gsu.edu/~tcpp/curriculum/?q=cedr_book

[2]This material is based upon work partially supported by the National Science Foundation under Grants IIS 1143533, CCF 1135124, CCF 1048711 and CNS 0950432. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

S. K. Prasad (✉)
Georgia State University, Atlanta, GA, USA
e-mail: sprasad@gsu.edu

A. Gupta
IBM Research AI, Yorktown Heights, NY, USA

A. L. Rosenberg · C. Weems
University of Massachusetts Amherst, Amherst, MA, USA

A. Sussman
University of Maryland, College Park, MD, USA

© Springer International Publishing AG, part of Springer Nature 2018

S. K. Prasad et al. (eds.), *Topics in Parallel and Distributed Computing*,
https://doi.org/10.1007/978-3-319-93109-8_1

## Why the CDER Book Project?

A curriculum working group drawn from the IEEE Technical Committee on Parallel Processing (TCPP), the National Science Foundation (NSF), and sibling communities such as the ACM and industry, has taken up the challenge of proposing and refining curricular guidelines for blending PDC-related concepts into early-stage undergraduate curricula in computational areas. A first version of the group's guidelines for a core curriculum that includes PDC was released in December 2012.[3] This curriculum and related activities – see Appendix for a brief description of the NSF/TCPP Curriculum Initiative – have spawned a vibrant international community of educators who are committed to PDC education. It is from this community that the desirability of the *CDER Book Project*, a series of books to support both instructors and students of PDC, became evident.

Curricular guidelines such as those promulgated by both us and the CS2013 ACM/IEEE Computer Science Curriculum Joint Task Force are an essential first step in propelling the teaching of PDC-related material into the twenty-first century. But such guidelines are only a first step: both instructors and students will benefit from suitable textual material to effectively translate guidelines into the curriculum. Moreover, experience to this point has made it clear that the members of the PDC community have much to share with each other and with aspiring new members, in terms of creativity in forging new directions and experience in evaluating existing ones. The Book Project's goal is to engage the community to address the need for suitable textbooks and related textual material to integrate PDC topics into the lower level core courses (which we affectionately, and hopefully transparently, refer to as CS1, CS2, Systems, Data Structures and Algorithms, Logic Design, etc.), and, as appropriate, into upper level courses. The current edited book series intends, over time, to cover all of these proposed topics.

In 2016, we invited proposals for chapters on teaching parallel and distributed computing topics, suitable for either instructors or students, specifically on topics from the current NSF/TCPP curriculum guidelines for introductory courses that have not been addressed by the chapters in the earlier book. Subsequently, we saw good community interest in authoring chapters for higher level elective courses as well. To address this, we extended the scope of this book to both lower-level core courses and more advanced, specialized topics in parallel and distributed computing that are targeted at students in upper level classes. The book has evolved organically based on contributions received in response to calls for book chapters. All contributions have been rigorously reviewed internally by the editors and externally by experts.

[3]Prasad, S. K., Chtchelkanova, A., Dehne, F., Gouda, M., Gupta, A., Jaja, J., Kant, K., La Salle, A., LeBlanc, R., Lumsdaine, A., Padua, D., Parashar, M., Prasanna, V., Robert, Y., Rosenberg, A., Sahni, S., Shirazi, B., Sussman, A., Weems, C., and Wu, J. 2012. NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing – Core Topics for Undergraduates, Version I, Online: https://grid.cs.gsu.edu/~tcpp/curriculum/, 55 pages.

## Book Organization

This book has two parts.

Part I – For instructors: These chapters are aimed at instructors to provide background, scholarly materials, insights into pedagogical strategies, and descriptions of experience with both strategies and materials. The emphasis is on the basic concepts and references on what and how to teach PDC topics in the context of the existing topics in various core courses.

Part 2 – For students: These chapters provide supplemental textual material for core courses that students can rely on for learning and exercises. These are envisioned as being at the quality of a textbook presentation, with many illustrations and examples, following a sequence of smaller steps to build larger concepts. We envision the student materials as supplemental sections that could be inserted into existing texts by instructors.

*Print and Free Web Publication:* While a print version through a renowned commercial publisher will foster our dissemination efforts in a professional format, the preprint versions of all the chapters of this book series will be freely available on the CDER Book Project website.[4]

*Chapter Organization:* This introductory chapter is organized as follows. Section "Chapter Introductions" gives brief outlines of each of the ten subsequent chapters. Section "How to Find a Topic or Material for a Course?" provides a roadmap for the readers to find suitable chapters and sections within these which are relevant for specific courses or PDC topics from the NSF/TCPP Curriculum. Section "Editor and Author Biographical Sketches" contains biographical sketches of the editors and authors. Appendix gives a brief history of the NSF/TCPP Curriculum Initiative.

## Chapter Introductions

### *Part I: For Instructors*

Chapter 2, *What Do We Need to Know about Parallel Algorithms and Their Efficient Implementation?,* by Vladimir Voevodin, Alexander Antonov, and Vadim Voevodin, explores a two-phase paradigm for teaching parallel algorithmics. A student is first taught to investigate an algorithmic problem in a machine-independent manner, learning to recognize opportunities for exploiting concurrency and to identify inherent sequentiality that will preclude such exploitation. After having mastered

[4]CDER Book Project – Free Preprint Version: http://cs.gsu.edu/~tcpp/curriculum/?q=CDER_Book_Project

this inherent aspect of the problem, the student is taught to investigate the problem in the context of a variety of computing platforms. This two-phase paradigm allows a student to approach computing with an understanding of the opportunities and challenges provided by the structure of the problem to be solved, as well as the opportunities and challenges provided by the structure of the computing platform one has access to. This chapter is intended for the instructors of introductory courses on parallel algorithms.

In chapter 3, titled *Modules for Teaching Parallel Performance Concepts*, Apan Qasem discusses three teaching modules targeting parallel performance concepts. The first module discusses fundamental concepts in parallel computing performance and mainly targets a CS1 course, highlighting parallel programming tools and performance metrics, and provides several sample exercises. The second module targets an upper-level operating systems class and focuses on communication and synchronization and how they affect performance for parallel applications, introducing the concepts of data dependences, synchronization, race conditions, and load balancing, again providing several sample exercises. The third module focuses on performance measurement and estimation of parallel systems, targeting compiler and computer architecture classes. This module reviews basic performance concepts and discusses advanced concepts such as strong and weak scaling, linear and super-linear speedup, and latency vs. bandwidth measurements in the context of OpenMP, and provides two sample exercises.

Chapter 4, *Scalability in Parallel Processing*, by Yanik Ngoko and Denis Trystram, provides a broad exposure to the notion of scalability, which is so important in modern (and future) large-scale parallel computing environments. The chapter discusses how scalability manifests itself and the many ways in which the "degree" of scalability is measured. The classical laws of Amdahl and Gustafson provide a central focus. The original arguments leading to those laws are described, accompanied by a reexamination of the laws' applicability in today's machines and computational problems. The notion of scalability is then further examined in the light of the evolution of the field of computing, with explorations of modern resource-sharing techniques and the more specific issue of reducing energy consumption. The chapter ends with a statistical approach to the design of scalable algorithms, specifically by organizing teams of parallel algorithms that "cooperate" in solving a single problem. The technically sophisticated aspects of organizing such cooperations is illustrated using the classical Satisfiability Problem. This chapter is intended for intermediate and advanced courses on the design and analysis of parallel algorithms.

In chapter 5, titled *Energy Efficiency Issues in Computing Systems*, Krishna Kant introduces energy efficiency issues in computer systems. Traditionally, computing has focused only on performance at all levels including circuits, architecture, algorithms, and systems. With power consumption and power density playing a central role at all these levels, it is crucial to teach students about power and performance tradeoffs. Power and energy issues are gaining importance in the context of mobile and embedded systems as well as server farms and data centers, although for different reasons. This chapter introduces topics like the basics of energy, power and thermal issues in computing, importance of and technology trends in power consumption, power-performance tradeoffs, power states, power adaptation, and energy efficiency of parallel programs.

Chapter 6, *Scheduling for fault-tolerance*, by Guillaume Aupy and Yves Robert, addresses a problem that has plagued large-scale parallel computing since its development in the 1970s and 1980s – fault tolerance. The electronically "aggressive" circuitry that enables high-performance large-scale parallel computing is vulnerable to both (permanent) failures and (transient) faults. Achieving high performance in practice, even for perfectly parallel applications, therefore demands the use of techniques that cope with these vulnerabilities. This chapter discusses the challenges of coping with faults and failures and introduces three simple strategies to achieve this: checkpointing, prediction, and work replication. Scheduling techniques are developed to optimize these three strategies. This chapter is intended for intermediate and advanced courses on the design and analysis of parallel algorithms. An operational understanding of elementary probability theory is necessary for true mastery of this material.

## *Part 2: For Students*

In chapter 7, titled *MapReduce – The Scalable Distributed Data Processing Solution*, Bushra Anjum provides students with an overview of how to process large datasets using the MapReduce programming model. Along with multiple examples of MapReduce applications, the chapter provides an outline of the basic functions that must be written to build a MapReduce application, and also discusses how the map and reduce steps in a distributed MapReduce system (i.e., Hadoop) execute a MapReduce application with scalable performance. The chapter also discusses the strengths and limitations of the MapReduce model, addressing scalability, flexibility, and fault tolerance. Finally, the chapter discusses higher level services built on top of the basic Hadoop MapReduce system.

In chapter 8, titled *The Realm of Graphical Processing Unit (GPU) Computing*, Vivek Pallipuram and Jinzhu Gao provide an introduction to general-purpose graphical processing unit (GPGPU) computing using the Compute Unified Device Architecture (CUDA) programming model. The chapter extensively covers the GPGPU architecture as viewed by a CUDA programmer and CUDA concepts including CUDA thread management, memory management, and performance optimization strategies. The chapter pedagogically reinforces the CUDA concepts using parallel patterns such as matrix-matrix multiplication and convolution. The chapter includes several active-learning exercises that engage students with the text. Throughout this chapter, students will develop an ability to write effective CUDA codes for maximum application performance. The chapter is intended for an upper-level undergraduate course with object-oriented programming and data structures using C++ as prerequisites. The chapter can also be used in a sophomore- or junior-level software engineering course, or in an undergraduate elective course dedicated to high-performance computing using a specialized architecture. Because the chapter covers the GPGPU architecture and programming in detail, a prior exposure to CS1/CS2 level programming with basic computer organization is desirable.

In chapter 9, titled *Managing Concurrency in Mobile User Interfaces with Examples in Android*, Konstantin Läufer and George K. Thiruvathukal discuss parallel and distributed computing from a mobile application development perspective, specifically addressing concurrency in interactive, GUI-based applications on the Android platform. The chapter gives an overview of GUI-based applications and frameworks, then looks at implementing simple interactive application behavior in the Android mobile application development framework using a running example. More complex use cases are introduced that enable discussing event handling and timers, to further show how GUI applications display all the benefits and costs of concurrent execution. Finally, the chapter closes with a deeper exploration of long-running compute-bound applications, where the problem is to maintain responsiveness to user requests.

In chapter 10, titled *Parallel Programming for Interactive GUI Applications*, Nasser Giacaman and Oliver Sinnen show students how to use Java threads to implement a graphical user interface (GUI) that is responsive even while computation is being done. Because this example of concurrency is concrete and visual, it can be introduced fairly early in the curriculum. If the first course in programming makes active use of the Java GUI framework, then this will be a modest extension of that coverage. By at least the second course in programming (again if GUI programming is already included), and certainly in a sophomore software engineering class, this material can be used as a means to introduce many ideas that are basic to PDC, and get students thinking in terms of using explicit concurrency to take advantage of the capabilities of modern systems. The foundation that is laid by this material could easily be extended, for example, in a programming with data structures course, to introduce thread-safe processing of larger structures, including algorithms such as parallel merge sort.

Chapter 11, titled *Scheduling in Parallel and Distributed Computing Systems* by Srishti Srivastava and Ioana Banicescu addresses the important topic of mapping tasks onto computational resources for parallel execution. The chapter provides an introduction to scheduling in PDC systems such that it can be understood by undergraduate students who are exposed to this topic for the first time. It contains detailed taxonomy of scheduling methods and comparisons between different methods from the point of view of applicability as well performance metrics such as runtime, speedup, efficiency, etc.

## How to Find a Topic or Material for a Course?

Table 1 lists the remaining chapters in the book, core/elective undergraduate courses they can be used for (see list below), and their prerequisites, if any. More detailed chapter-wise tables which follow list the topics covered in each chapter.

### *Relevant Courses and Prerequisites*

CORE COURSES:

| | |
|---|---|
| CS0: | Computer Literacy for Non-majors |
| CS1: | Introduction to Computer Programming (First Course) |
| CS2: | Second Programming Course in the Introductory Sequence |
| Systems: | Introductory Systems/Architecture Course |
| DS/A: | Data Structures and Algorithms |
| CE1: | Digital Logic (First Course) |

ADVANCED/ELECTIVE COURSES:

| | |
|---|---|
| Arch2: | Advanced Elective Course on Architecture |
| Algo2: | Elective/Advanced Algorithm Design and Analysis (CS7) |
| Lang: | Programming Language/Principles (after introductory sequence) |
| SwEngg: | Software Engineering |
| ParAlgo: | Parallel Algorithms |
| ParProg: | Parallel Programming |
| Compilers: | Compiler Design |
| Networking: | Communication Networks |
| DistSystems: | Distributed Systems |
| OS: | Operating Systems |

### *Chapters and Topics*

The following tables list the topics covered in each chapter. The depth of coverage of each topic is indicated by the intended outcome of teaching that topic, expressed using Bloom's taxonomy of educational objectives:

| | |
|---|---|
| K | = Know the term |
| C | = Comprehend so as to paraphrase/illustrate |
| A | = Apply it in some way |

**Table 1** Relevant Courses and Prerequisites

| Chap # | Short title | Primary core course | Other courses | Prerequisites |
|---|---|---|---|---|
| | | Part I | | |
| 2 | Parallel algorithms and implementation | CS0 | CS1, DS/A, ParAlgo | – |
| 3 | Parallel performance concepts | CS1, Systems | OS, DS/A | DS/A for advanced modules |
| 4 | Scalability | Systems | CS2, ParAlgo | Math maturity |
| 5 | Energy efficiency | CE1 | CS2, DS/A | Math maturity |
| 6 | Scheduling for fault tolerance | Systems | CS2, DS/A | CS1, Probabilities |
| | | Part II | | |
| 7 | MapReduce | DS/A | CS2, ParAlgo, DistSystems | CS0, CS1 |
| 8 | GPU computing | Systems | Arch 2, ParProg | CS1, CS2 |
| 9 | Mobile user interfaces | DS/A | Lang, ParProg | CS1, CS2 |
| 10 | Interactive GUI applications | CS2 DS/A | SwEng, ParProg | CS1, CS2 |
| 11 | Scheduling | DS/A | ParAlgo, DistSystems | Basic PDC terms and concepts |

## Editor and Author Biographical Sketches

### *Editors*

*Anshul Gupta* is a Principal Research Staff Member in IBM Research AI at IBM T.J. Watson Research Center. His research interests include sparse matrix computations and their applications in optimization and computational sciences, parallel algorithms, and graph/combinatorial algorithms for scientific computing. He has coauthored several journal articles and conference papers on these topics and a textbook titled "Introduction to Parallel Computing." He is the primary author of Watson Sparse Matrix Package (WSMP), one of the most robust and scalable parallel direct solvers for large sparse systems of linear equations.

*Sushil K. Prasad* (BTech'85 IIT Kharagpur, MS'86 Washington State, Pullman; PhD'90 Central Florida, Orlando – all in Computer Science/Engineering) is a Professor of Computer Science at Georgia State University and Director of Distributed and Mobile Systems (DiMoS) Lab. Sushil has been honored as an ACM Distinguished Scientist in Fall 2013 for his research on parallel data structures and applications. He was the elected chair of IEEE Technical Committee on Parallel Processing for two terms (2007–2011), and received its

Chapter 2: What Do We Need to Know About Parallel Algorithms and Their Efficient Implementation?

| PDC concept | Chapter section | | |
|---|---|---|---|
| | 2.1 | 2.2 | 2.3 |
| Performance issues | C | C | C |
| Information structure | C | C | C |
| Data locality | C | C | |
| Computational intensity | K | | |
| Resource of parallelism | C | C | C |
| Computational kernel | | K | |
| Serial complexity | | K | C |
| Parallel complexity | | K | C |
| Load balancing | | C | A |
| Determinacy of an algorithm | | C | |
| Scalability | | C | A |
| Efficiency | | C | C |
| Race conditions | | | A |

Chapter 3: Modules for Teaching Parallel Performance Concepts

| PDC concept | Chapter section | | |
|---|---|---|---|
| | 3.2 | 3.3 | 3.4 |
| Speedup | K | | C |
| Efficiency | K | | C |
| Linear and super linear speedup | K | | C |
| Strong and weak scaling | K | | C |
| Amdahl's law | C | | A |
| Power vs. time trade-offs | K | | A |
| Task granularity | | A | |
| Load balancing | | A | |
| Communication and synchronization | | C | |
| Scheduling and thread mapping | | A | |
| SMP and NUMA | | | C |
| Data locality | | | C |

highest honors in 2012 – IEEE TCPP Outstanding Service Award. Currently, he is leading the NSF-supported IEEE-TCPP curriculum initiative on parallel and distributed computing with a vision to ensure that all computer science and engineering graduates are well-prepared in parallelism through their core courses in this era of multi- and many-cores desktops and handhelds. His current research interests are in Parallel Data Structures and Algorithms, and Computation over Geo-Spatiotemporal Datasets over Cloud, GPU and Multicore Platforms. Sushil is currently a Program Director leading the Office of Advanced Cyberinfrastructure (OAC) Learning and Workforce Development crosscutting

Chapter 4: Scalability in Parallel Processing

| PDC concept | Chapter section | | | | |
|---|---|---|---|---|---|
| | 4.1 | 4.2 | 4.3 | 4.4 | 4.5 |
| Scalability | K | C | | | |
| Speedup | | C | C | C | A |
| Efficiency | | C | C | C | A |
| Data parallelism | | | K | | |
| Isoefficiency | | C | | | |
| Amdahl' law | | K | C | C | A |
| Gustafson' law | | K | C | C | A |
| Strong scaling | | C | | | C |
| Weak scaling | | C | | | C |
| Resource sharing | | | | C | A |
| Energy efficiency | | K | | K | |
| P-completeness | | | K | | |
| Algorithm portfolio | | | | | A |

Chapter 5: Energy Efficiency Issues in Computing Systems

| PDC concept | Chapter section | | | | | | |
|---|---|---|---|---|---|---|---|
| | 5.1 | 5.2 | 5.3 | 5.4 | 5.5 | 5.6 | 5.7 |
| Energy efficiency in computing | C | C | | | | | |
| Power states and their Management | | | K | K | | | |
| Software energy efficiency | | | | | K | | |
| Energy efficiency vs. parallelism | | | | | | C | |
| Energy adaptation | | | | | | | N |

Chapter 6: Scheduling for Fault-Tolerance

| PDC concept | Chapter section | | | | | |
|---|---|---|---|---|---|---|
| | 6.1 | 6.2 | 6.3 | 6.4 | 6.5 | 6.6 |
| Why/What is par/Dist computing | K | K | K | K | K | K |
| Performance issues, Computation | K | A | C | A | A | K |
| Cluster | K | K | K | K | K | K |
| Performance measures | | A | A | A | A | |
| Basic probabilities | | C | A | C | C | |
| Programming SPMD | | C | | | | |
| Load balancing | | K | | K | A | |
| Scheduling | | C | | C | C | |
| Dynamic programming | | | | | A | |

programs at U.S. National Science Foundation. His homepage is www.cs.gsu.edu/prasad.

*Arnold L. Rosenberg* holds the rank of Distinguished University Professor Emeritus in the School of Computer Science at the University of Massachusetts Amherst.

Chapter 7: MapReduce: The Scalable Distributed Data Processing Solution

| PDC concept | Chapter section | | | | |
|---|---|---|---|---|---|
| | 7.1 | 7.2 | 7.3 | 7.4 | 7.5 |
| Why/What is par/Dist computing | A | A | C | K | A |
| Concurrency | K | K | C | | A |
| Cluster computing | A | C | A | K | K |
| Scalability | A | C | A | K | A |
| Speedup | K | C | A | | |
| Divide & Conquer (parallel aspects) | C | A | K | | A |
| Recursion (parallel aspects) | | | K | | A |
| Scan (parallel-prefix) | K | A | | | C |
| Reduction (map-reduce) | K | A | | K | A |
| Time | C | A | | | A |
| Sorting | K | A | | | A |

Chapter 8: The Realm of Graphical Processing Unit (GPU) Computing

| PDC concept | Chapter section | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 8.1 | 8.2 | 8.3 | 8.4 | 8.5 | 8.6 | 8.7 | 8.8 | 8.9 |
| Data parallelism | C | | | | | | | | |
| GPGPU devices | | C | A | A | A | A | A | A | |
| nvcc compiler | | | A | | | | | | |
| Thread management | | | | A | A | | | | |
| Parallel patterns | | | | | | | A | A | |
| Performance evaluation | | | | | | | A | | |
| Performance optimization | | | | | | | A | A | |
| CUDA | | A | | A | A | A | A | A | |
| Advancements in GPU computing | | | | | | | | | K |

Prior to joining UMass, Rosenberg was a Professor of Computer Science at Duke University from 1981 to 1986, and a Research Staff Member at the IBM Watson Research Center from 1965 to 1981. He has held visiting positions at Yale University and the University of Toronto, as well as research professorships at Colorado State University and Northeastern University. He was a Lady Davis Visiting Professor at the Technion (Israel Institute of Technology) in 1994, and a Fulbright Senior Research Scholar at the University of Paris-South in 2000. Rosenberg's research focuses on developing algorithmic models and techniques to exploit the new modalities of "collaborative computing" (wherein multiple computers cooperate to solve a computational problem) that result from emerging computing technologies. Rosenberg is the author or coauthor of more than 190 technical papers on these and other topics in theoretical computer science and discrete mathematics. He is the coauthor of the research book *Graph Separators, with Applications* and the author of the textbook *The Pillars of Computation*

Chapter 9: Managing Concurrency in Mobile User Interfaces with Examples in Android

| PDC concept | Chapter section | | | | | | |
|---|---|---|---|---|---|---|---|
| | 9.1 | 9.2 | 9.3 | 9.4 | 9.5 | 9.6 | 9.7 |
| Why and what is PDC | K | | | | | | |
| Tasks and threads | K | | C | | | A | A |
| Thread safety | | K | C | C | A | A | A |
| Race conditions | | K | C | C | C | A | A |
| Thread/Task spawning | | | K | | | A | A |
| Synchronization | | | C | C | A | A | A |
| Nondeterminism | | | C | | | A | |
| Deadlocks | | | | K | | | |

Chapter 10: Parallel Programming for Interactive GUI Applications

| PDC concept | Chapter section | | | | | |
|---|---|---|---|---|---|---|
| | 10.1 | 10.2 | 10.3 | 10.4 | 10.5 | 10.6 |
| Concurrency | C | A | A | C | A | A |
| Race conditions | | C | A | | | |
| Thread safety | | | C | C | | |
| GUI concurrency | | | | C | A | A |

*Theory: State, Encoding, Nondeterminism*; additionally, he has served as coeditor of several books. Rosenberg is a Life Fellow of the ACM, a Life Fellow of the IEEE, a Golden Core member of the IEEE Computer Society, and a member of the Sigma Xi Research Society. Rosenberg received an A.B. in mathematics at Harvard College and an A.M. and Ph.D. in applied mathematics at Harvard University.

*Alan Sussman* is a Professor in the Department of Computer Science and Institute for Advanced Computer Studies at the University of Maryland. Working with students and other researchers at Maryland and other institutions he has published numerous conference and journal papers and received several best paper awards in various topics related to software tools for high performance parallel and distributed computing, and has contributed chapters to six books. His research interests include peer-to-peer distributed systems, software engineering for high performance computing, and large scale data intensive computing. Software tools he has built with his graduate students have been widely distributed and used in many computational science applications, in areas such as earth science, space science, and medical informatics. He is a subject area editor for the Parallel Computing journal and an associate editor for IEEE Transactions on Services Computing, and edited a previous book on teaching parallel and distributed computing. He is a founding member of the Center for Parallel and Distributed Computing Curriculum Development and Educational Resources (CDER). He received his Ph.D. in computer science from Carnegie Mellon University.

Chapter 11: Scheduling in Parallel and Distributed Computing Systems

| PDC concept | Chapter section | | | |
|---|---|---|---|---|
| | 11.1 | 11.2 | 11.3 | 11.4 |
| MIMD architecture | K | | | |
| Multicore | C | | | |
| SMP | N | | | C |
| Topologies | | N | N | |
| Latency | | K | | |
| Heterogeneous | | K | | K |
| Data Parallel | | C | | |
| Computation | C | C | C | C |
| Load balancing | | C | C | C |
| Distributed memory | | | C | C |
| Client server | | | C | |
| Static | | C | C | |
| Dynamic | | C | C | C |
| Asymptotic | C | | | |
| Communication | | C | C | |
| Synchronization | | C | C | |
| Speedup | | | A | |
| Efficiency | | A | A | |
| Makespan | | C | | C |
| Concurrency | | C | | |
| Performance modeling | | | | K |
| Fault tolerance | K | | | |

*Charles Weems* is co-director of the Architecture and Language Implementation lab at the University of Massachusetts. His current research interests include architectures for media and embedded applications, GPU computing, and high precision arithmetic, and he has over 100 conference and journal publications. Previously he led development of two generations of a heterogeneous parallel processor for machine vision, called the Image Understanding Architecture, and co-directed initial work on the Scale compiler that was eventually used for the TRIPS architecture. He is the author of numerous articles, has served on many program committees, chaired the 1997 IEEE CAMP Workshop, the 1999 IEEE Frontiers Symposium, co-chaired IEEE IPDPS in 1999, 2000, and 2013, was general vice-chair for IPDPS from 2001 through 2005, is on the steering committees of EduPar and EduHPC. He has co-authored 28 introductory CS texts. He is a member of ACM, Senior Member of IEEE, a member of the Executive Committee of the IEEE TC on Parallel Processing, has been an editor for IEEE TPDS, Elsevier JPDC, and is an editor with Parallel Computing.

## Authors

*Bushra Anjum* has a PhD in Computer Science from North Carolina State University and is currently serving as a Tech Lead for the Amazon Prime Program at Amazon, Inc. Alongside, she is also a visiting professor at the Computer Science Department of the California Polytechnic Institute, San Luis Obispo. Anjum has been extensively using Elastic MapReduce platform provided by Amazon Web Services for a few years now for job related tasks. Before joining industry, she served in academia full time both in the USA and in Pakistan for 5+ years. With unconventional career choices and international exposure, she brings the expertise of being an academician, a researcher and a practitioner at the same time.

*Alexander Antonov* is a leading researcher in Research Computing Center of Lomonosov Moscow State University (RCC MSU). His main research interests are related to research in such fields as parallel and distributed computing, performance and efficiency of computers, parallel programming, informational structure of algorithms and programs, application optimization and fine tuning, architecture of computers, benchmarks, etc. In 1999 Alexander Antonov received PhD degree on the subject of interprocedural analysis of programs. Alexander took part in a number of projects supported by the Ministry of Education and Sciences of the Russian Federation, Russian Foundation for Basic Research and Russian Science Foundation. Alexander is editor of Parallel.Ru Information analytical center for parallel computing. Alexander Antonov is one of the main developers of the AlgoWiki Open encyclopedia of parallel algorithmic features. At the present time Alexander Antonov takes part in different researches being conducted in RCC MSU that are devoted to efficiency analysis of parallel applications and supercomputer systems in general. He has published over 50 scientific papers with 4 books among them.

*Guillaume Aupy* received his PhD from ENS Lyon. He is currently a tenured researcher at Inria Bordeaux Sud-Ouest. His research interests include data-aware scheduling, reliability, energy efficiency in high-performance computing. He is the author of numerous articles, has served on many program committees. He was the technical program vice-chair of SC17.

*Ioana Banicescu* is a professor in the Department of Computer Science and Engineering at Mississippi State University (MSU). Between 2009 and 2017, she was also a Director of the NSF Center for Cloud and Autonomic Computing at MSU. Professor Banicescu received the Diploma in Electronics and Telecommunications from Polytechnic University – Bucharest, and the M.S. and the Ph.D. degrees in Computer Science from New York University – Polytechnic Institute. Her research interests include parallel algorithms, scientific computing, scheduling and load balancing algorithms, performance modeling, analysis and prediction, and autonomic computing. Between 2004–2017, she was an Associate Editor of the Cluster Computing journal and the International Journal on Computational Science and Engineering. Professor Banicescu, served and continues to serve on numerous research review panels for advanced research grants in the US and Europe, on steering and program committees of a number of international conferences, symposia and workshops, on the Executive Board and Advisory Board of the IEEE Technical Committee on Parallel Processing (TCPP). She has given many invited talks at universities, government laboratories, and at various national and international forums in the United States and overseas.

*Jinzhu Gao* received the Ph.D. degree in Computer Science from The Ohio State University in 2004. From June 2004 to August 2008, she worked at the Oak Ridge National Laboratory as a research associate and then the University of Minnesota, Morris, as an Assistant Professor of Computer Science. She joined the University of the Pacific (Pacific) in 2008 and is currently a Professor of Computer Science at Pacific. Her main research focus is on intelligent data visual analytics. Over the past 15 years, Dr. Gao has been working closely with application scientists and Silicon Valley technology companies to develop online data visual analytics and deep learning platforms to support collaborative science, mobile health, IoT data analytics, business operational visibility, and visual predicative analysis for industries. Her work has been published in top journals such as IEEE Transactions on Visualization and Computer Graphics, IEEE Transactions on Computers, and IEEE Computer Graphics and Applications.

*Nasser Giacaman* is a Senior Lecturer in the Department of Electrical and Computer Engineering at the University of Auckland, New Zealand. His disciplinary research interest includes parallel programming, particularly focusing on high-level languages in the context of desktop and mobile applications running on multi-core systems. He also researches Software Engineering Education by driving the development of tools and apps to help students learn difficult programming concepts.

*Krishna Kant* is a full professor in the department of computer and information science at Temple University. He has 37 years of combined experience in academia, industry, and government and has published in a wide variety of areas in computer science, telecommunications, and logistics systems. His current research interests span energy management, data centers, wireless networks, resilience in high performance computing, critical infrastructure security, storage systems, database systems, configuration management, and logistics networks. He is a fellow of IEEE.

*Konstantin Läufer* is a full professor of computer science at Loyola University Chicago. He received a PhD in computer science from the Courant Institute at New York University in 1992. His research interests include programming languages, software architecture, and distributed and pervasive computing systems. His recent focus in research and teaching has been on the impact of programming languages, methodologies, frameworks, and tools on software quality. Konstantin has repeatedly served as a consultant in academia and industry and is a co-inventor on two patents owned by Lucent Technologies.

*Yanik Ngoko* received his B.Sc. in Computer Science from University of Yaoundé I (UYI), Cameroon, his M.Sc. in parallel and numerical computing also from UYI, and his doctorate in Computer Science from the Institut National Polytechnique de Grenoble, France (2010). From 2011 to 2014, he was a postdoctoral researcher, first at the university of São Paulo and then at the university of Paris 13. Since October 2014, he is a research scientist at Qarnot computing and an associate member of the Laboratoire d'Informatique de Paris Nord (University of Paris 13). His research interests include parallel and distributed computing, web services, cloud computing, applications of edge computing to IoT.

*Vivek Pallipuram* (B.Tech.2008 NIT Trichy, MS2010 Clemson University, Ph.D.2013 Clemson University) is an Assistant Professor of Computer Engineering at University of the Pacific, Stockton, California. His research interests include high-performance computing (HPC), heterogeneous architectures such as general-purpose graphical processing units (GPGPUs) and Xeon Phi co-processors, Cloud computing, image processing, and random signal processing. His interests also include promoting HPC education and scientific computing in primarily-undergraduate universities. His work has been published in journals such as the Journal of Supercomputing, and Concurrency and Computation: Practice and Experience; and in top conferences such as IEEE Cluster and eScience. He is also a peer-reviewer for a number of international journals and conference proceedings. In the classroom, he strives to be a facilitator by engaging students using active-learning techniques. In addition to receiving information from the instructor, students interact with their peers via in-class group activities and gain valuable perspective. This process increases the influx of knowledge per student, promoting well-rounded and comprehensive learning. He enjoys teaching high-performance computing, computer systems and networks, random signals, and image processing.

*Apan Qasem* is an Associate Professor in the Computer Science Department at Texas State University. He received his PhD in 2008 from Rice University. Qasem directs the Compilers Research Group at Texas State where he and his students are working on a number of projects in the area of high-performance computing including developing intelligent software for improving programmer productivity and using GPUs for general-purpose computation. Qasem's research has received funding from the National Science Foundation, Department of Energy, Semiconductor Research Consortium (SRC), IBM, Nvidia and the Research Enhancement Program at Texas State. In 2012, he received an NSF CAREER award to pursue research in autotuning of exascale systems. Qasem has co-authored over 50 peer-reviewed publications including two that won best paper awards. He regularly teaches the undergraduate and graduate Compilers and Computer Architecture courses.

*Yves Robert* received the PhD degree from Institut National Polytechnique de Grenoble. He is currently a full professor in the Computer Science Laboratory LIP at ENS Lyon. He is the author of 7 books, 150 papers published in international journals, and 240 papers published in international conferences. He is the editor of 11 book proceedings and 13 journal special issues. He is the advisor of 30 PhD theses. His main research interests are scheduling techniques and resilient algorithms for large-scale platforms. He is a Fellow of the IEEE. He has been elected a Senior Member of Institut Universitaire de France in 2007 and renewed in 2012. He has been awarded the 2014 IEEE TCSC Award for Excellence in Scalable Computing, and the 2016 IEEE TCPP Outstanding Service Award. He holds a Visiting Scientist position at the University of Tennessee Knoxville since 2011.

*Oliver Sinnen* graduated in Electrical and Computer Engineering at RWTH Aachen University, Germany. Subsequently, he moved to Portugal, where he received his PhD from Instituto Superior Técnico (IST), University of Lisbon, Portugal in 2003. Since 2004 he is a (Senior) Lecturer in the Department of Electrical and Computer Engineering at the University of Auckland, New Zealand, where he leads the Parallel and Reconfigurable Computing Lab. His research interests include parallel computing and programming, scheduling and reconfigurable computing. Oliver authored the book "Task Scheduling for Parallel Systems", published by Wiley.

*Srishti Srivastava* is an Assistant Professor of Computer Science at the University of Southern Indiana. She received her Ph.D. in Computer Science at Mississippi State University in May 2015. Her research interests include dynamic load balancing in parallel and distributed computing, performance modeling, optimization, and prediction, robustness analysis of resource allocations, and autonomic computing. Srishti has authored and co-authored a number of articles published in renowned IEEE and ACM conferences, journals, and book chapters. Srishti has served on the program committees of international conference workshops such as, EduHPC, and EduPar. She has also been a peer reviewer for a number of international journals, and conference proceedings. She is a professional member of the IEEE computer society, ACM, Society for Industrial and Applied Mathematics (SIAM), Computing Research Association (CRA, CRA-W), Anita Borg Institute Grace Hopper Celebration (ABI-GHC), and an honor society of Upsilon Pi Epsilon (UPE). She is also a 2014 young researcher alumna of the Heidelberg Laureate Forum, Germany.

*George K. Thiruvathukal* received his PhD from the Illinois Institute of Technology in 1995. He is a full professor of computer science at Loyola University Chicago and visiting faculty at Argonne National Laboratory in the Mathematics and Computer Science Division, where he collaborates in high-performance distributed systems and data science. He is the author of three books, co-editor of a peer-reviewed collection, and author of various peer-reviewed journal and conference papers. His early research involved object-oriented approaches to parallel programming and the development of object models, languages, libraries, and tools (messaging middleware) for parallel programming, mostly based on C/C++ on Unix platforms. His subsequent work in Java resulted in the book *High-Performance Java Platform Computing*, Prentice Hall and Sun Microsystems Press, 2000. He also co-authored the book *Codename Revolution: The Nintendo Wii Platform* in the MIT Press Platform Studies Series, 2012. Recently, he coedited *Software Engineering for Science*, Taylor and Francis/CRC Press, October 2016.

*Denis Trystram* is a Professor in Computer Science at Grenoble Institute of technology since 1991 and is now distinguished professor there. He was a senior member of Institut Universitaire de France from 2010 to 2014. He obtained in 2011 a Google research award in Optimization for his contributions in the field of multi-objective Optimisation. Denis is leading a research group on optimization of resource management for parallel and distributed computing platforms in a joint team with Inria. Since 2010, he is director of the international Master program in Computer Science at university Grenoble-Alpes. He has been elected recently as the director of the research pole in Maths and Computer Science in this university.

*Vadim Voevodin* is a senior research fellow in Research computing center of Lomonosov Moscow state university (RCC MSU). His main research interests are related to different aspects of high-performance computing: analysis of parallel program efficiency, development of system software, parallel programming, etc. Vadim Voevodin got his PhD in memory locality analysis in parallel computing. Also he was a main developer in a research devoted to the study of memory hierarchy usage. At the present time Vadim Voevodin is actively involved in different researches being conducted in RCC MSU that are devoted to efficiency analysis of parallel applications and supercomputer systems in general. One research is dedicated to detecting abnormal inefficient job behavior based on constant monitoring of supercomputer job flow. The other newly started research is aimed to develop a universal software tool suite that will help common users to conduct both large-scale efficiency analysis of the entire set of applications and a professional in-depth analysis of individual parallel applications, based on many researches previously done in RCC MSU. Another major research area concerns the analysis of supercomputer resource utilization and efficiency of using application packages installed on a supercomputer.

*Vladimir Voevodin* is Deputy Director of the Research Computing Center at Lomonosov Moscow State University. He is Head of the Department "Supercomputers and Quantum Informatics" at the Computational Mathematics and Cybernetics Faculty of MSU, professor, corresponding member of Russian academy of sciences. Vl. Voevodin specializes in parallel computing, supercomputing, extreme computing, program tuning and optimization, fine structure of algorithms and programs, parallel programming technologies, scalability and efficiency of supercomputers and applications, supercomputing co-design technologies, software tools for parallel computers, and supercomputing education. His research, experience and knowledge became a basis for the supercomputing center of Moscow State University, which was founded in 1999 and is currently the largest supercomputing center in Russia. He has contributed to the design and implementation of the following tools, software packages, systems and online resources: V-Ray, X-Com, AGORA, Parallel.ru, hpc-education.ru, hpc-russia.ru, LINEAL, Sigma, Top50, OctoShell, Octotron, AlgoWiki. He has published over 100 scientific papers with 4 books among them. Voevodin is one of the founders of Supercomputing Consortium of Russian Universities established in 2008, which currently comprises more than 60 members. He is a leader of the major national activities on Supercomputing Education in Russia and General Chair of the two largest Russian supercomputing conferences.

## Appendix: A Brief History of The NSF/TCPP Curriculum Initiative

The pervasiveness of computing devices containing multicore CPUs and GPUs, including PCs, laptops, tablets, and mobile devices, is making even casual users of computing technology beneficiaries of parallel processing. Certainly, technology has developed to the point where it is no longer sufficient for even basic programmers to acquire only the sequential programming skills that are the staple in computing curricula. The trends in technology point to the need for imparting a broad-based skill set in PDC technology at various levels in the educational fabric woven by Computer Science and Computer Engineering programs as well as their allied computational disciplines. To address this need, a curriculum working group drawn from the IEEE Technical Committee on Parallel Processing (TCPP), the National Science Foundation (NSF), and sibling communities such as the ACM and industry, has taken up the challenge of proposing and refining a curricular guidleines for blending PDC-related concepts into even early-stage undergraduate curricula in computational areas. This working group is built around a constant core of members and typically includes members from all segments of the computing world and the geographical world. A first version of the group's guidelines for a core curriculum that includes PDC was released informally in December, 2010, with a formal version[3] following in December 2012. The CS2013 ACM/IEEE Computer Science Curriculum Joint Task Force has recognized the need to integrate parallel and distributed computing topics in the early core courses in the computer science and computer engineering curriculum, and has collaborated with our working group in leveraging our curricular guidelines. The CS2013 curriculum[5] explicitly refers to the NSF/TCPP curricular guideines for comprehensive coverage of parallelism (and provides a direct hyperlink to the guidelines).

The enthusiastic reception of the CDER guidelines has led to a commitment within the working group to continue to develop the guidelines and to foster their adoption at an even broader range of academic institutions. Toward these ends, the Center for Curriculum Development and Educational Resources (CDER) was founded, with the five editors of this volume comprising the initial Board of Directors. An expanded version of the working group has taken up the task of

[5]The ACM/IEEE Computer Science Curricula 2013: (https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf)

revising and expanding the 2012 NSF/TCPP curriculum during the 2016–2018 timeframe. One avenue for expansion has been to add special foci on a select set of important aspects of computing that are of particular interest today – Big Data, Energy-Aware Computing, Distributed Computing – and to develop Exemplars that will assist instructors in assimilating the guidelines' suggested topics into their curricula. CDER has initiated several activities toward the goal of fostering PDC education.

1. A *courseware repository*[6] has been established for pedagogical materials – sample lectures, recommended problem sets, experiential anecdotes, evaluations, papers, etc. This is a living repository. CDER invites the community to contribute existing and new material to it. The Exemplars aspect group is working to provide extensive set of exemplars for various topics and courses.
2. An *Early Adopter Program* has been established to foster the adoption and evaluation of the guidelines. This activity has fostered educational work on PDC at more than 100 educational institutions in North and South America, Europe, and Asia. The Program has thereby played a major role in establishing a worldwide community of people interested in developing and implementing PDC curricula. Additional early adopter training workshops and competitions are planned.
3. The *EduPar workshop series* has been established. The original instantiation of EduPar was as a satellite of the International Parallel and Distributed Processing Symposium (IPDPS). EduPar was – and continues to be – the first education-oriented workshop at a major research conference. The success of EduPar led to the development of a sibling workshop, EduHPC, at the Supercomputing Conference (SC) in 2013. In 2015 EduPar and EduHPC was joined by a third sibling workshop, Euro-EduPar, a satellite of the International Conference on Parallel Computing (EuroPar). CDER has also sponsored panels, and BOF and special sessions at the ACM Conference on Computer Science Education (SIGCSE).
4. A *CDER Compute Cluster* has been setup for free accesses by the early adopters and other educators and their students. The CDER cluster is a heterogeneous 14-node cluster featuring 280 cores, 1 TB of RAM, and GPUs that are able to sustain a mixed user workload.[7]

[6]CDER Courseware Repository: https://grid.cs.gsu.edu/~tcpp/curriculum/?q=courseware_management

[7]CDER Cluster free access: https://grid.cs.gsu.edu/~tcpp/curriculum/?q=node/21615