计 算 机 科 学 丛 书

Parallel Computer Architecture

A Hardware/Software Approach

并行计算机体系结构

(英文版·第2版)

(美) David E. Culler
Jaswinder Pal Singh
Anoop Gupta 著

机械工业出版社 China Machine Press
MORGAN KAUFMANN

计算机科学丛书

并行计算机体系结构

（英文版 第2版）

Parallel Computer Architecture A Hardware/Software Approach

(Second Edition)

David E. Culler

(美) Jaswinder Pal Singh 著

Anoop Gupta

机械工业出版社
China Machine Press

本书版权登记号：图字：01-1999-2021

图书在版编目（CIP）数据

并行计算机体系结构：第2版：英文/(美)勒斯(Culler, D. E.)等著，-北京：机械工业出版社，1999.9

(计算机科学丛书)

ISBN 7-111-07440-8

Ⅰ.并… Ⅱ.勒… Ⅲ.并行计算机－系统结构－英文 Ⅳ.TP338.6

中国版本图书馆CIP数据核字(1999)第33790号

出 版 者：马九荣（北京市西城区百万庄大街22号 邮政编码 100037）

北京市密云县印刷厂印刷 · 新华书店北京发行所发行

1999年9月第1版第1次印刷

787mm × 1092mm1/16 · 65.75印张

印数：0 001- 2 000册

定价：88.00元

To Sara, Silvia, and our families

In memory of Susanne Kreith Culler

About the Authors

David E. Culler, *Professor of Computer Science at University of California, Berkeley.* Dr. Culler works in the areas of computer architecture, communication, programming languages, operating systems, and performance analysis. He led the Berkeley Network of Workstations (NOW) project, which sparked the current commercial revolution in high-performance clusters. He is internationally known for his work on Active Messages for fast communication, the LogP parallel performance model, the Split-C parallel language, the TAM threaded abstract machine, and for his work on dataflow architectures. He received the Presidential Faculty Fellowship Award and the Presidential Young Investigator Award from the National Science Foundation. He received the PhD from MIT in 1989. Currently he is vice-chair of computing and networking for the Deptartment of Electrical Engineering and Computer Sciences at UC Berkeley and leads the Millennium Project, investigating campus-wide clusters.

Jaswinder Pal Singh, *Assistant Professor in the Computer Science Department at Princeton University.* Dr. Singh works at the boundary of parallel applications and multiprocessor systems, including architecture, software, and performance evaluation. He has led the development and distribution of the SPLASH and SPLASH-2 suites of parallel programs, which are very widely used in parallel systems research. While at Stanford, where he obtained his MS and PhD degrees, he participated in the DASH and FLASH multiprocessor projects, leading the applications efforts there. The technology developed in the DASH project is becoming widely available in commercial products. At Princeton, he heads PRISM, an application-driven research group that investigates supporting programming models on a variety of communication architectures and applies parallel computing to a variety of application domains. He is a recipient of the Presidential Early Career Award for Scientists and Engineers (PECASE) and a Sloan Research Fellowship.

Anoop Gupta, *Associate Professor of Computer Science and Electrical Engineering at Stanford University and Senior Researcher at Microsoft.* Dr. Gupta has worked in the areas of computer architecture, operating systems, programming languages, performance debugging tools, and parallel applications. With John Hennessy, he co-led the design and construction of the Stanford DASH machine, one of the first scalable distributed shared memory multiprocessors, and has worked on the follow-up FLASH project. The technology developed in the DASH project is now becoming widely available in commercial products. Professor Gupta has published close to 100 papers in major conferences and journals, including several award papers. Professor Gupta received the NSF Presidential Young Investigator Award and held the Robert Noyce faculty scholar chair at Stanford. He obtained the PhD from Carnegie Mellon University in 1986.

Foreword

John L. Hennessy
Frederick Emmons Terman Dean of Engineering, Stanford University

I am delighted to be able to write the foreword for this exciting and timely new book on parallel computing. The insightful approach taken by the authors combined with a systematic and quantitative examination of different architectures distinguishes this book from all previous books on parallel architecture. The approach, which is developed in the first four chapters, has three major innovations: it builds on the recent convergence of parallel architectures, it uses applications as a driver for evaluating and analyzing architectures, and it is grounded in a solid methodology for performance evaluation.

The recent convergence among the shared memory and message-passing paradigms, which is described in Chapter 1, provides new opportunities for characterizing and analyzing architectures in a common framework. Relying on this convergence, the authors describe four fundamental design issues (communication abstraction, programming model, communication and replication, and performance) that create a framework for talking about a wide variety of architectures and implementations. Within this framework, different architectural approaches are compared and examined critically.

One cannot understand the design trade-offs or performance of multiprocessors without understanding the interaction of applications and architecture. Accordingly, Chapters 2 and 3 describe a set of parallel programs as well as how the applications are parallelized and organized for performance. These chapters illuminate both the parallel programming process and its challenges in addition to laying a foundation for quantitative evaluation of architectural approaches and implementations. These chapters are key to understanding the performance of multiprocessors, and Chapter 4 illustrates this by showing how to evaluate an architecture using a parallel workload. The authors also describe the complexities of evaluating parallel machines, including issues arising from the scaling of machine sizes and workloads. Together these three chapters form the foundation on which the remaining chapters build.

Small-to-medium-sized shared memory multiprocessors are the dominant form of parallel architecture seen today, and understanding the principles and design trade-offs of these machines is critical to anyone interested in parallel computing.

Chapter 5 describes the key concepts underlying shared memory multiprocessing: cache coherency, memory consistency, and synchronization. The authors then describe the detailed design of snoop-based shared memory multiprocessors, including two detailed case studies, in Chapter 6.

Designing multiprocessors that scale to larger numbers of processing nodes remains one of the most challenging and controversial aspects of multiprocessor architecture. Chapter 7 devotes itself to such machines, spanning the design space from message passing to shared memory. Chapter 8 extends this discussion by examining the use of directory schemes, which allow cache coherency to scale to larger numbers of processing nodes. The basics of directory-based coherence are discussed, and two detailed case studies form the core of the chapter. These case studies are the first detailed and quantitative examinations of commercial implementations of directory-based cache coherence.

Some of the most important hardware and software technologies used in multiprocessors are largely independent of the details of the architectural approach. Hence, the authors explore these key technologies in a set of three chapters. Chapter 9 describes the software implications, hardware requirements, and performance trade-offs that arise in memory systems, including both consistency issues and the extended use of caching. Chapter 10 examines interconnection technology, a key constituent of any multiprocessor. Finally, Chapter 11 examines techniques for tolerating latency, in many ways the key "universal" design problem for parallel computers.

The book concludes with an insightful discussion of future hardware and software challenges. First, the authors discuss likely evolutionary scenarios in the hardware and software domain. Then they turn to the potential hurdles in a pair of sections entitled "Hitting a wall." Finally, they examine potential breakthroughs! I found the final chapter both stimulating and thought provoking. The different backgrounds and complementary strengths of the authors help make this chapter both perspicacious and provocative.

In summary, this is an exciting and dynamic new exploration of the multiprocessor design space. The convergence in architectural approaches combined with the authors' framework has made it possible to establish a common ground on which to examine the diversity of modern parallel architectures. A few years ago, it would have been impossible to write this book because the architectural approaches were too divergent. Similarly, without the attention to quantitative measures of performance and the interaction between applications and architectures, this book would be much less distinctive. Instead, the authors have taken advantage of the convergence and the focus on an applications-driven and performance-based analysis to produce a unique and insightful exploration of parallel architectures. This approach, combined with the unique strengths and experiences of the authors, yields a treatise that is far more perceptive than any other book in parallel architecture. I congratulate the authors and commend this book to all readers interested in both the practice and concepts of parallel processing and the future of these technologies.

Preface

Parallel computing has become a critical component of the computing technology of the 1990s, and it is likely to have as much impact over the next 20 years as microprocessors have had over the past 20. Indeed, the two technologies are deeply linked, as the evolution of highly integrated microprocessors and memory chips makes multiprocessor systems increasingly attractive. Multiprocessors already represent the high-performance end of almost every segment of the computing market, from the fastest supercomputers and largest data centers to departmental servers to the individual desktop. Tightly integrated clusters of PCs, workstations, or even multiprocessors are emerging as scalable Internet servers. In the past, computer vendors employed a range of technologies and processor architectures to provide increasing performance across their product line. Today, the same state-of-the-art microprocessor is used throughout. To obtain a significant range of performance, the primary approach is to increase the number of processors, and the economies of scale make this extremely attractive. Very soon, several processors will fit on a single chip and multiprocessors will be even more widespread than they are today.

Although parallel computing has a long and rich academic history, the close coupling with commodity technology has fundamentally changed the discipline. The emphasis on radical architectures and exotic technology has given way to quantitative analysis, the realization of different programming models on the same underlying processing nodes, and careful engineering trade-offs. Our goal in writing this book is to equip designers of the emerging class of multiprocessor systems—from modestly parallel desktop computers to highly parallel information servers and supercomputers—with an understanding of the fundamental architectural and software issues and the available techniques for addressing design trade-offs. At the same time, we hope to provide designers of software systems and applications with an understanding of the likely directions of architectural evolution, the forces that will determine the specific path that hardware designs will follow, and the impact of these developments on performance-oriented programming.

The most exciting recent development in parallel computer architecture is the convergence of traditionally disparate approaches—namely, shared memory, message-passing, data parallel, and data-driven computing—on a common machine structure. This convergence is driven partly by common technological and economic forces and partly by a better understanding of parallel software. It allows us to develop a common framework in which to understand and evaluate architectural trade-offs rather than to focus on exotic designs and taxonomies. Moreover, popular

parallel programming models are available on a wide range of machines, making parallel programming more portable and allowing meaningful benchmarks and evaluation methodologies to flourish. This maturing of the field makes it possible to undertake a quantitative as well as qualitative study of hardware/software interactions. In fact, it demands such an approach. The book follows a set of issues that are critical to all parallel architectures—data access, communication performance, coordination of cooperative work, and correct implementation of useful semantics—across the full range of modern designs. It describes the set of techniques available in hardware and in software to address each issue and explores how the various techniques interact. Carefully chosen, in-depth case studies provide a concrete illustration of the general principles and demonstrate specific interactions between mechanisms.

One of the motivations for writing this book is the lack of an adequate textbook for our own courses at Berkeley, Princeton, and Stanford. Several existing texts cover the material in a cursory fashion, summarizing various architectures and research results but not analyzing them in depth or providing a modern engineering framework. Others focus on specific projects but do not carry the principles over to alternative approaches. The research reports in the area provide a sizable body of ideas and empirical data, but it is not distilled into a coherent picture. By focusing on the salient issues in the context of technological and architectural convergence rather than on the rich and varied history that has brought us to this point, we hope to provide a deeper and more coherent understanding of this exciting and rapidly changing field. This was a deeply collaborative effort, reflected in the alternation of the order of our names on the book covers.

Intended Audience

The subject matter of this book is core material that is important for researchers, students, and practicing engineers in the fields of computer architecture, systems software, and applications. The relevance for computer architects is obvious, given the growing importance of multiprocessors. Chip designers must understand what constitutes a viable building block for multiprocessor systems. Bus and memory system design are dominated by issues related to parallelism. I/O system design must address fast scalable networks, clustering, and devices that are shared by multiple processors.

Systems software—including operating systems, compilers, programming languages, run-time systems, and performance debugging tools—needs to address new issues and will provide new opportunities in parallel computers. Thus, an understanding of architectural evolution and the forces guiding that evolution is critical. Research and development in compilers and programming languages have addressed aspects of parallel computing for some time. However, the new convergence with commodity technology suggests that these aspects may need to be reexamined and addressed in a more general context. The traditional boundaries between hardware, operating system, and user program are also shifting in the context of parallel

computing, where programs often want more direct control over resources for better performance.

Applications areas, such as computer graphics and multimedia, scientific computing, computer-aided design, databases, decision support, and transaction processing, are all likely to see a tremendous transformation as a result of the vast computing power available at low cost through parallel computing. However, developing parallel applications that are robust and that provide good parallel speedup across current and future multiprocessors is a challenging task and requires a deep understanding of system interactions and architectural directions. The book seeks to provide this understanding but also to stimulate the exchange between the applications fields and computer architecture so that better architectures can be designed—those that make the programming task easier and performance both higher and more robust.

Organization of the Book

The book is organized into 12 chapters. Chapter 1 provides an overview of parallel architecture. It opens with a discussion of why the expanding role of multiprocessors is inevitable, given current trends in technology, architecture, and applications. It briefly introduces the diverse multiprocessor architectures that have shaped the field (shared memory, message passing, data parallel, dataflow, and systolic) and shows how the technology and architectural trends are driving a convergence in the field to a set of commodity processing nodes connected by a communication architecture. This convergence does not mean the end to innovation but, on the contrary, that we will now see a time of rapid progress, as designers start talking with each other rather than past each other. The chapter develops a layered framework (including the programming model, communication abstraction, user/system interface, and hardware/software interface) for understanding wide variety of communication architectures and implementations. Viewing the convergence of the field in this framework, the last portion of the chapter lays out the fundamental design issues that must be addressed at each of the interfaces between layers: naming, ordering, replication, and communication performance (overhead, latency, and bandwidth). These issues form an underlying theme throughout the rest of this book. The chapter ends with a set of historical references.

Chapter 2 provides an introduction to the process of parallel programming. It describes a set of motivating applications for multiprocessors that are used throughout the rest of the book. It shows what parallel programs look like in the major programming models and hence what primitives a system must support. It uses the application case studies to illustrate the steps of decomposition, assignment, orchestration, and mapping in creating a parallel program and identifies the key performance goals of these steps.

Chapter 3 describes the basic techniques that good parallel programmers use to get performance out of the underlying architecture. It provides an understanding of hardware/software trade-offs and illustrates what aspects of performance can be addressed through architectural means and what aspects must be addressed either

by the compiler or the programmer. The analogy in sequential computing is that architecture cannot transform an $O(n^2)$ algorithm into an $O(n \log n)$ algorithm, but it can improve the average access time for common memory reference patterns. The chapter shows clearly the core algorithmic and programming challenges that cut across programming models as well as the model-specific orchestration issues. This material shows how architectural advance can ease the burden of effective parallel programming in addition to increasing the achievable performance. The programming techniques are a key factor in any quantitative evaluation of design trade-offs, and the chapter concludes by applying them to the motivating applications to produce high-performance versions.

Chapter 4 takes up the challenge of performing solid workload-driven evaluation of design trade-offs. Architectural evaluation is difficult even for modern uniprocessors, where we typically look at moderate design variations—such as pipeline or memory system organizations—against a fixed set of programs. In parallel architecture, we have many more degrees of freedom to explore. The interactions between aspects of the design are more profound, and the interactions between hardware and software are more significant as well as of wider scope. We are often interested in performance as the machine and the program scale, and it is impossible to scale one without affecting the other. It is easy to arrive at incomplete or even misleading conclusions if the evaluation is not methodologically sound, so the characteristics of parallel programs must be adequately understood. Chapter 4 discusses how application and architectural parameters interact and how they should be scaled together and presents benchmarks that are used throughout later chapters. It provides methodological guidelines for the evaluation of real machines and of architectural ideas through simulation. The Appendix provides additional reference material on parallel benchmarking efforts.

Chapters 5 and 6 provide a complete understanding of the bus-based, symmetric shared memory multiprocessors (SMPs) that form the bread and butter of modern commercial machines beyond the desktop. Chapter 5 presents the high-level, logical design of "snooping" bus protocols, which ensure that automatically replicated data is coherent across multiple caches. This chapter provides an important discussion of memory consistency, which brings us to terms with what shared memory really means to algorithm designers. It discusses the spectrum of design options and how machines are optimized against typical reference patterns occurring in user programs and in the operating system. Given this conceptual understanding of SMPs, the chapter reflects on the implications for parallel software, including applications and support for synchronization.

Chapter 6 examines the protocol issues in more depth as well as physical design of bus-based multiprocessors. It digs into the engineering issues that arise in supporting modern microprocessors with multilevel caches on modern buses, which are highly pipelined, as well as how the high-level protocols of the previous chapter are realized and extended on these systems. The presentation here provides a very complete understanding of the design issues in this regime. It is all the more important because these small-scale designs form a building block for large-scale designs and because many of the concepts appear later in the book on a larger scale with a

broader set of concerns. The chapter also provides self-contained case studies on the SGI Challenge and Sun Enterprise servers.

Chapters 7, 8, 9, and 10 provide a complete understanding of the scalable multiprocessor architectures that represent the high end of computing and the future of the midrange as technology continues to advance.

Chapter 7 presents the hardware organization and architecture of a range of machines that are scalable to large or very large configurations. The key organizational concept is that of a network transaction, analogous to the bus transaction that is the fundamental primitive for the smaller designs in Chapters 5 and 6. However, in scalable machines the global arbitration and globally visible information is lost and a large number of transactions can be outstanding. The chapter shows how programming models are realized in terms of network transactions and studies a spectrum of important design points organized according to the level of direct hardware interpretation of the network transaction, including case studies of the nCUBE/2, Thinking Machines CM-5, Intel Paragon, Meiko CS-2, CRAY T3D, and CRAY T3E. It examines modern clusters in this framework with case studies of the Myrinet NOW and the DEC Memory Channel. A performance comparison is conducted across these designs.

Chapter 8 puts the results of the previous chapters together to demonstrate how to realize a shared physical address space with automatic hardware replication and cache coherence on scalable systems. This style of machine is increasingly popular in the industry. The chapter provides a complete treatment of directory-based cache coherence protocols and hardware design alternatives, including case studies of the SGI Origin2000 and Sequent NUMA-Q. It examines workload behavior on these machines and extends the discussions of programming implications and synchronization.

Chapter 9 examines a spectrum of alternatives for shared address space systems that push the boundaries of hardware/software trade-offs to obtain higher performance, reduce hardware cost and complexity, or both. It covers relaxed memory consistency models, cache-only memory architectures that replicate data coherently in hardware in main memory, and software-based coherent replication. Much of this material is in the transitional phase from academic research to commercial product at the time of this writing, and its role will be further shaped as cluster technology emerges. It exposes very important design concepts not treated elsewhere in the book.

Chapter 10 addresses the design of scalable high-performance communication networks, which underlies all the scalable machines discussed in previous chapters but was deferred to complete our understanding of the processor, memory system, and network interface design that drive these networks. The chapter builds a general framework for understanding where hardware costs, transfer delays, and bandwidth restrictions arise in networks. It looks at a variety of trade-offs in routing techniques, switch design, and interconnection topology with respect to these cost-performance metrics. The trade-offs are made concrete through case studies of recent designs.

Given the foundation established by the first 10 chapters, Chapter 11 examines a set of crosscutting issues involved in tolerating the significant latencies that arise in

multiprocessor systems without impeding performance. The techniques exploit two basic capabilities: overlapping latency with useful work and pipelining the transfer of data. The simplest of these techniques are essentially bulk transfers, which pipeline the movement of a large regular sequence of data items and often can be offloaded from the processor. The other techniques attempt to hide the latency incurred in collections of individual loads and stores. Write latencies are hidden by exploiting weak consistency models, which recognize that ordering is conveyed by only a small set of the accesses to shared memory in a program. Read latencies are hidden by implicit or explicit prefetching of data or by lookahead techniques in modern dynamically scheduled processors. Some of the techniques extend to hiding synchronization latencies as well. The chapter provides a thorough examination of these alternatives, the impact on compilation techniques, and a quantitative evaluation of effectiveness.

Finally, Chapter 12 examines the trends in technology, architecture, software systems, and applications that are likely to shape the future evolution of the field. It looks at evolutionary scenarios, walls we may hit, and potential breakthroughs from a hardware/software perspective.

Using the Book

The book is organized to meet the needs of several potential audiences. It can serve as a graduate text, a professional reference for engineers, and as a general reference for members of the technical community who find themselves dealing ever more frequently with parallel computing. There is sufficient material, if covered in full depth, for a full-year study of parallel computing, covering the entire range of machine design and practical parallel programming experience. However, it can also be used in smaller segments.

Chapter 1 is intended to provide a stand-alone, general understanding of parallel architectures as would be appropriate for a segment of a general computer architecture course at the graduate or upper-division undergraduate level. It would also be appropriate for the engineering manager or corporate executive needing to understand the vocabulary and basic concepts of parallel computing and how the technology will impact their business. It lays out clearly where to go to learn more as your interest or need to understand parallel computing increases. The chapter can also be used as a basic background in parallel architecture for compiler, database, operating system, or programming courses. Chapters 1 and 12 together provide a well-rounded "outer skin" of parallel computer architecture.

A parallel architecture course oriented toward machine organization and design is comprised of the core material of Chapters 5, 6, 7, 8, and 10, in addition to the overview of Chapter 1. However, the chapters go into greater depth of design than has been common in traditional courses because the material was not available in any published form or put together in a design-oriented framework, and they provide detailed quantitative illustrations of trade-offs. Chapters 5 and 6 develop the key requirements of correctness in cache-coherent systems and show how to satisfy them with high performance in increasingly complex designs. Chapter 7 takes apart

scalable machines in a manner not available from commercial sources or research publications and addresses emerging high-performance clusters in this framework. Chapter 8 describes the cache coherence protocols of prominent commercial distributed-memory machines in a framework and level of detail not available elsewhere. Chapter 10 provides a compact, rounded treatment of network design. The treatment is deep enough in these chapters to provide even the seasoned system designer with a new understanding and a clean design framework. A serious yet pragmatic treatment of memory consistency models is carried throughout these chapters (as well as in the first part of Chapter 9), as is a discussion of implementing synchronization operations. These chapters on machine organization and design can be supplemented with Chapter 11, which covers the increasingly important topic of latency tolerance.

The exciting opportunity presented by this text is that, with the core material packaged in a cohesive form, it becomes possible to strengthen the basic parallel architecture course along several dimensions. First, thorough coverage of Chapters 2 and 3 allows the treatment to reach across the hardware/software boundary. This gives the architecture student a much more solid grasp of the impact of architectural decisions and what parallel programming is all about. It also broadens the appeal of the course to a wider audience of operating systems, languages, and applications students who are viewing the architectural issues from a software perspective. A second dimension along which the basic course can be strengthened is quantitative performance analysis of hardware and software design decisions. Building upon a basic understanding from Chapters 2 and 3, Chapter 4, the Appendix, and the "Implications for Parallel Software" sections of the later chapters carry this thread throughout the core machine design material. They provide an informed, critical perspective with which to view published results, as well as methodological guidelines for performing evaluations. A third dimension is a sharp focus on hardware/software tradeoffs. This is the underlying issue that is framed by the quantitative analysis and explored in the synchronization and programming sections of each chapter. It comes to the fore in Chapter 9, where the division of responsibilities in providing a coherent shared address space is examined in detail, and in Chapter 11 in the discussion of latency tolerance. Each of these dimensions represents a group of professionals who have an increasing need to understand more deeply how to deal with parallel architectures.

The book also serves well as the primary text for a hands-on parallel programming course. With Chapter 1 providing a general introduction, Chapters 2 and 3 offer a strong framework for how to reason about the behavior of parallel programs. This is further solidified by the workload analysis in Chapter 4 and the "Implications for Parallel Software" sections in Chapters 5, 7, 8, and 9. This material should be supplemented with a reference on the parallel programming environment used in the course, such as MPI, parallel threads, or HPF. The case studies in Chapters 6, 7, and 8 provide thorough coverage of machines similar to what students are likely to use. Chapter 11 provides a convenient framework for an examination of how best to solve the challenges of communication in parallel programming.

We believe parallel computer architecture is an exciting core field of study and practice whose importance will continue to grow. It has reached a point of maturity at which a serious textbook based on design and engineering principles makes sense. From a rich diversity of ideas and approaches, a dramatic convergence is now occurring in the field. It is time to go beyond surveying the machine landscape to an understanding of the fundamental design principles. We have intimately participated in the convergence of the field; this text arises from our experience, and we hope it conveys some of the excitement that we feel for this dynamic and growing area. Since parallel architecture does change so rapidly, case studies, performance analyses, and workloads need to be refreshed periodically. The Web page for this book will provide a repository for such timely material, as well as for additional teaching materials, and we hope that you will help contribute to that repository through the high-quality products of your courses and commercial developments. The URL for the book is *www.mkp.com/pca*.

We also encourage readers to report any errors or bugs so that we may correct them in subsequent printings. Please email them to *pcabugs@mkp.com*. Please also check the errata page at *www.mkp.com/pca* to see if the bug has already been reported and fixed.

Acknowledgments

This book has been in gestation in various forms for quite some time, and it has benefited from the efforts of many individuals. It had its roots in notes and slides for our parallel processing courses and in our research projects. Our students and staff have been invaluable throughout. Although this is the first edition, drafts have been available on the Web as the material was being developed. In the way of the Web, we have no idea of all the institutions around the world that have used it in courses and research, but we receive suggestions from the most exotic places. Many people have made contributions to it directly, indirectly, or even anonymously, so we would like to thank all of you.

Numerous students have improved this book by their questions, ideas, solutions, and projects. We want to thank the students in CS 258 (Parallel Processors) and CS 267 (Applications of Parallel Computers) at Berkeley, CS 598 (Parallel Computer Architecture and Programming) at Princeton, and CS 315A (Parallel Computer Architecture and Programming) and CS 315B (Parallel Programming Project) at Stanford. Special thanks go to Andrea Arpaci-Dusseau, Remzi Arpaci-Dusseau, Brent Chun, Seth Goldstein, Alan Mainwaring, Rich Martin, Lok Tin Liu, Steve Lummetta, Chad Yoshikawa, and Frederick Chun Bong Wong at Berkeley; Angelos Bilas, Liviu Iftode, Dongming Jiang, Steven Kleinstein, Sanjeev Kumar, Hongzhang Shan, and Yuanyuan Zhou at Princeton; and Cheng Chen, John Heinlein, Moriyoshi Ohara, Evan Torrie, and Steven Cameron Woo at Stanford, all of whom have contributed valuable insight, data, and analysis to the book through their tireless efforts. Jiang, Kumar, Ohara, Torrie, Wong, and Woo deserve an especially hearty thanks for their contributions.

Many people in academia and industry provided invaluable assistance in reviewing drafts, explaining to us how things really worked, trying out the book, and guiding us along the path. We would especially like to thank Sarita Adve, Arvind, Russell Clapp, Michel Dubois, Mike Galles, Kourosh Gharachorloo, Jim Gray, John Hennessy, Mark Hill, Phil Krueger, James Laudon, Edward Lazowska, Dan Lenoski, W. R. Michalson, Todd Mowry, Greg Papadopoulos, Dave Patterson, Randy Rettberg, Shuichi Sakai, Klaus Schauser, Ashok Singhal, Burton Smith, Jim Smith, Mark Smotherman, Per Stenstrom, Thorsten von Eicken, Maurice Wilkes, David Wood, and Chengzhong Xu. Thanks, John and Dave, for guidance throughout. Many people assisted us by teaching from portions of the book, including our earliest adopters, Sarita Adve, Andrew Chien, Jim Demmel, Wallid Najjar, Constantine Polychronopoulos, Radhika Thekkath, and Kathy Yelick.

We also want to thank the National Science Foundation, the Defense Advanced Research Projects Agency, the Department of Energy, and numerous corporate sponsors for supporting the research that underlies the material in this book and the dramatic advance of parallel computing.

We wish to thank the impressive team at Morgan Kaufmann Publishers who managed to get this book all the way to the end. Denise Penrose picked up the reins and led the team with unbelievable energy, dedication, and enthusiasm. It was an absolute pleasure to work with her. Elisabeth Beller managed the entire production process very smoothly. Meghan Keeffe and Jane Elliott coordinated reviews and photo searches and tied up many a loose end. A crew of talented proofreaders kept all the right words in all the right places. Thanks also to Jennifer Mann, who managed the project before Denise joined up, and to Bruce Spatz, who has moved on from MKP since starting this book on its way.

We must also thank our university staff, Gabriela Aranda, Ginny Hogan, Chris Kranz, Terry Lessard-Smith, Bob Miller, Thoi Nguyen, Matt Norcross, Charlie Orgish, Jim Roberts, and Chris Tengi, for countless bits of help along the way.

Above all, our deepest thanks, appreciation, and love go to our families for their immeasurable support, patience, kindness, and wisdom throughout the entire process.

Contents