

Parallel Computer Architecture A Hardware Software Approach

Although multicore is now a mainstream architecture, there are few textbooks that cover parallel multicore architectures. Filling this gap, *Fundamentals of Parallel Multicore Architecture* provides all the material for a graduate or senior undergraduate course that focuses on the architecture of multicore processors. The book is also useful as a reference. The Second Edition of *The Cache Memory Book* introduces systems designers to the concepts behind cache design. The book teaches the basic cache concepts and more exotic techniques. It leads readers through some of the most intricate protocols used in complex multiprocessor caches. Written in an accessible, informal style, this text demystifies cache memory design by translating cache concepts and jargon into practical methodologies and real-life examples. It also provides adequate detail to serve as a reference book for ongoing work in cache memory design. The Second Edition includes an updated and expanded glossary of cache memory terms and buzzwords. The book provides new real world applications of cache memory design and a new chapter on cache "tricks". Illustrates detailed example designs of caches Provides numerous examples in the form of block diagrams, timing waveforms, state tables, and code traces Defines and discusses more than 240 cache specific buzzwords, comparing in detail the relative merits of different design methodologies Includes an extensive glossary, complete with clear definitions, synonyms, and references to the appropriate text discussions

Innovations in hardware architecture, like hyper-threading or multicore processors, mean that parallel computing resources are available for inexpensive desktop computers. In only a few years, many standard software products will be based on concepts of parallel programming implemented on such hardware, and the range of applications will be much broader than that of scientific computing, up to now the main application area for parallel computing. Rauber and Runger take up these recent developments in processor architecture by giving detailed descriptions of parallel programming techniques that are necessary for developing efficient programs for multicore processors as well as for parallel cluster systems and supercomputers. Their book is structured in three main parts, covering all areas of parallel computing: the architecture of parallel systems, parallel programming models and environments, and the implementation of efficient application algorithms. The emphasis lies on parallel programming techniques needed for different architectures. For this second edition, all chapters have been carefully revised. The chapter on architecture of parallel systems has been updated considerably, with a greater emphasis on the architecture of multicore systems and adding new material on the latest developments in computer architecture. Lastly, a completely new chapter on general-purpose GPUs and the corresponding programming techniques has been added. The main goal of the book is to present parallel programming techniques that can be used in many situations for a broad range of application areas and which enable the reader to develop correct and efficient parallel programs. Many examples and exercises are provided to show how to apply the techniques. The book can be used as both a textbook for students and a reference book for professionals. The material presented has been used for courses in parallel programming at different universities for many years.

Essential reading to understand patterns for parallel programming Software patterns have revolutionized the way we think about how software is designed, built, and documented, and the design of parallel software requires you to consider other particular design aspects and special skills. From clusters to supercomputers, success heavily depends on the design skills of software developers. *Patterns for Parallel Software Design* presents a pattern-oriented software architecture approach to parallel software design. This approach is not a design method in the classic sense, but a new way of managing and exploiting existing design knowledge for designing parallel programs. Moreover, such approaches enhance not only build-time properties of parallel systems, but also, and particularly, their run-time properties. Features known solutions in concurrent and distributed programming, applied to the development of parallel programs Provides architectural patterns that describe how to divide an algorithm and/or data to find a suitable partition and link it with a programming structure that allows for such a division Presents an architectural point of view and explains the development of parallel software *Patterns for Parallel Software Design* will give you the skills you need to develop parallel software.

High Performance Computing is an integrated computing environment for solving large-scale computational demanding problems in science, engineering and business. Newly emerging areas of HPC applications include medical sciences, transportation, financial operations and advanced human-computer interface such as virtual reality. High performance computing includes computer hardware, software, algorithms, programming tools and environments, plus visualization. The book addresses several of these key components of high performance technology and contains descriptions of the state-of-the-art computer architectures, programming and software tools and innovative applications of parallel computers. In addition, the book includes papers on heterogeneous network-based computing systems and scalability of parallel systems. The reader will find information and data relative to the two main thrusts of high performance computing: the absolute computational performance and that of providing the most cost effective and affordable computing for science, industry and business. The book is recommended for technical as well as management oriented individuals.

The performance of software systems is dramatically affected by how well software designers understand the basic hardware technologies at work in a system. Similarly, hardware designers must understand the far-reaching effects their design decisions have on software applications. For readers in either category, this classic introduction to the field provides a look deep into the computer. It demonstrates the relationships between the software and hardware and focuses on the foundational concepts that are the basis for current computer design.

Programming Massively Parallel Processors: A Hands-on Approach, Second Edition, teaches students how to program massively parallel processors. It offers a detailed discussion of various techniques for constructing parallel programs.

Case studies are used to demonstrate the development process, which begins with computational thinking and ends with effective and efficient parallel programs. This guide shows both student and professional alike the basic concepts of parallel programming and GPU architecture. Topics of performance, floating-point format, parallel patterns, and dynamic parallelism are covered in depth. This revised edition contains more parallel programming examples, commonly-used libraries such as Thrust, and explanations of the latest tools. It also provides new coverage of CUDA 5.0, improved performance, enhanced development tools, increased hardware support, and more; increased coverage of related technology, OpenCL and new material on algorithm patterns, GPU clusters, host programming, and data parallelism; and two new case studies (on MRI reconstruction and molecular visualization) that explore the latest applications of CUDA and GPUs for scientific research and high-performance computing. This book should be a valuable resource for advanced students, software engineers, programmers, and hardware engineers. New coverage of CUDA 5.0, improved performance, enhanced development tools, increased hardware support, and more Increased coverage of related technology, OpenCL and new material on algorithm patterns, GPU clusters, host programming, and data parallelism Two new case studies (on MRI reconstruction and molecular visualization) explore the latest applications of CUDA and GPUs for scientific research and high-performance computing

This book outlines a set of issues that are critical to all of parallel architecture--communication latency, communication bandwidth, and coordination of cooperative work (across modern designs). It describes the set of techniques available in hardware and in software to address each issues and explore how the various techniques interact.

Designed for introductory parallel computing courses at the advanced undergraduate or beginning graduate level, *Elements of Parallel Computing* presents the fundamental concepts of parallel computing not from the point of view of hardware, but from a more abstract view of algorithmic and implementation patterns. The aim is to facilitate the teaching of parallel programming by surveying some key algorithmic structures and programming models, together with an abstract representation of the underlying hardware. The presentation is friendly and informal. The content of the book is language neutral, using pseudocode that represents common programming language models. The first five chapters present core concepts in parallel computing. SIMD, shared memory, and distributed memory machine models are covered, along with a brief discussion of what their execution models look like. The book also discusses decomposition as a fundamental activity in parallel algorithmic design, starting with a naive example, and continuing with a discussion of some key algorithmic structures. Important programming models are presented in depth, as well as important concepts of performance analysis, including work-depth analysis of task graphs, communication analysis of distributed memory algorithms, key performance metrics, and a discussion of barriers to obtaining good performance. The second part of the book presents three case studies that reinforce the concepts of the earlier chapters. One feature of these chapters is to contrast different solutions to the same problem, using select problems that aren't discussed frequently in parallel computing textbooks. They include the Single Source Shortest Path Problem, the Eikonal equation, and a classical computational geometry problem: computation of the two-dimensional convex hull. After presenting the problem and sequential algorithms, each chapter first discusses the sources of parallelism then surveys parallel algorithms.

The Architecture of Computer Hardware, Systems Software and Networking is designed help students majoring in information technology (IT) and information systems (IS) understand the structure and operation of computers and computer-based devices. Requiring only basic computer skills, this accessible textbook introduces the basic principles of system architecture and explores current technological practices and trends using clear, easy-to-understand language. Throughout the text, numerous relatable examples, subject-specific illustrations, and in-depth case studies reinforce key learning points and show students how important concepts are applied in the real world. This fully-updated sixth edition features a wealth of new and revised content that reflects today's technological landscape. Organized into five parts, the book first explains the role of the computer in information systems and provides an overview of its components. Subsequent sections discuss the representation of data in the computer, hardware architecture and operational concepts, the basics of computer networking, system software and operating systems, and various interconnected systems and components. Students are introduced to the material using ideas already familiar to them, allowing them to gradually build upon what they have learned without being overwhelmed and develop a deeper knowledge of computer architecture.

Parallel Computations focuses on parallel computation, with emphasis on algorithms used in a variety of numerical and physical applications and for many different types of parallel computers. Topics covered range from vectorization of fast Fourier transforms (FFTs) and of the incomplete Cholesky conjugate gradient (ICCG) algorithm on the Cray-1 to calculation of table lookups and piecewise functions. Single tridiagonal linear systems and vectorized computation of reactive flow are also discussed. Comprised of 13 chapters, this volume begins by classifying parallel computers and describing techniques for performing matrix operations on them. The reader is then introduced to FFTs and the tridiagonal linear system as well as the ICCG method. Different versions of the conjugate gradient method for solving the time-dependent diffusion equation are considered. Subsequent chapters deal with two- and three-dimensional fluid flow calculations, paying particular attention to the principal issues in designing efficient numerical methods for hydrodynamic calculations; the decisions that a numerical modeler must make to optimize chemically reactive flow simulations; and how to handle disk-to-core data transfer and storage allocation for the solution of the implicit equations for three-dimensional flows. The book also describes the time-split finite difference scheme for solving the two-dimensional Navier-Stokes equation for flows through slotted nozzles. Finally, the large-scale stimulation of plasmas, as carried out on a small computer with an array processor, is discussed. This monograph should be of interest to specialists in computer science.

There is a software gap between the hardware potential and the performance that can be attained using today's software parallel program development tools. The tools need manual intervention by the programmer to parallelize the code. Programming a parallel computer requires closely studying the target algorithm or application, more so than in the traditional sequential programming we have all learned. The programmer must be aware of the communication and data dependencies of the algorithm or application. This book provides the techniques to explore the possible ways to program a parallel computer for a given application.

The new RISC-V Edition of *Computer Organization and Design* features the RISC-V open source instruction set architecture, the first open source architecture designed to be used in modern computing environments such as cloud computing, mobile devices,

and other embedded systems. With the post-PC era now upon us, Computer Organization and Design moves forward to explore this generational change with examples, exercises, and material highlighting the emergence of mobile computing and the Cloud. Updated content featuring tablet computers, Cloud infrastructure, and the x86 (cloud computing) and ARM (mobile computing devices) architectures is included. An online companion Web site provides advanced content for further study, appendices, glossary, references, and recommended reading. Features RISC-V, the first such architecture designed to be used in modern computing environments, such as cloud computing, mobile devices, and other embedded systems Includes relevant examples, exercises, and material highlighting the emergence of mobile computing and the cloud

This historical survey of parallel processing from 1980 to 2020 is a follow-up to the authors' 1981 Tutorial on Parallel Processing, which covered the state of the art in hardware, programming languages, and applications. Here, we cover the evolution of the field since 1980 in: parallel computers, ranging from the Cyber 205 to clusters now approaching an exaflop, to multicore microprocessors, and Graphic Processing Units (GPUs) in commodity personal devices; parallel programming notations such as OpenMP, MPI message passing, and CUDA streaming notation; and seven parallel applications, such as finite element analysis and computer vision. Some things that looked like they would be major trends in 1981, such as big Single Instruction Multiple Data arrays disappeared for some time but have been revived recently in deep neural network processors. There are now major trends that did not exist in 1980, such as GPUs, distributed memory machines, and parallel processing in nearly every commodity device. This book is intended for those that already have some knowledge of parallel processing today and want to learn about the history of the three areas. In parallel hardware, every major parallel architecture type from 1980 has scaled-up in performance and scaled-out into commodity microprocessors and GPUs, so that every personal and embedded device is a parallel processor. There has been a confluence of parallel architecture types into hybrid parallel systems. Much of the impetus for change has been Moore's Law, but as clock speed increases have stopped and feature size decreases have slowed down, there has been increased demand on parallel processing to continue performance gains. In programming notations and compilers, we observe that the roots of today's programming notations existed before 1980. And that, through a great deal of research, the most widely used programming notations today, although the result of much broadening of these roots, remain close to target system architectures allowing the programmer to almost explicitly use the target's parallelism to the best of their ability. The parallel versions of applications directly or indirectly impact nearly everyone, computer expert or not, and parallelism has brought about major breakthroughs in numerous application areas. Seven parallel applications are studied in this book.

Still Image Compression on Parallel Computer Architectures investigates the application of parallel-processing techniques to digital image compression. Digital image compression is used to reduce the number of bits required to store an image in computer memory and/or transmit it over a communication link. Over the past decade advancements in technology have spawned many applications of digital imaging, such as photo videotex, desktop publishing, graphics arts, color facsimile, newspaper wire phototransmission and medical imaging. For many other contemporary applications, such as distributed multimedia systems, rapid transmission of images is necessary. Dollar cost as well as time cost of transmission and storage tend to be directly proportional to the volume of data. Therefore, application of digital image compression techniques becomes necessary to minimize costs. A number of digital image compression algorithms have been developed and standardized. With the success of these algorithms, research effort is now directed towards improving implementation techniques. The Joint Photographic Experts Group (JPEG) and Motion Photographic Experts Group (MPEG) are international organizations which have developed digital image compression standards. Hardware (VLSI chips) which implement the JPEG image compression algorithm are available. Such hardware is specific to image compression only and cannot be used for other image processing applications. A flexible means of implementing digital image compression algorithms is still required. An obvious method of processing different imaging applications on general purpose hardware platforms is to develop software implementations. JPEG uses an 8×8 block of image samples as the basic element for compression. These blocks are processed sequentially. There is always the possibility of having similar blocks in a given image. If similar blocks in an image are located, then repeated compression of these blocks is not necessary. By locating similar blocks in the image, the speed of compression can be increased and the size of the compressed image can be reduced. Based on this concept an enhancement to the JPEG algorithm is proposed, called Block Comparator Technique (BCT). Still Image Compression on Parallel Computer Architectures is designed for advanced students and practitioners of computer science. This comprehensive reference provides a foundation for understanding digital image compression techniques and parallel computer architectures.

Parallel Computer Vision

An In-Depth View of Hardware Issues, Programming Practices, and Implementation of Key Methods Exploring the challenges of parallel programming from the perspective of quantum chemists, Parallel Computing in Quantum Chemistry thoroughly covers topics relevant to designing and implementing parallel quantum chemistry programs. Focusing on good parallel program design and performance analysis, the first part of the book deals with parallel computer architectures and parallel computing concepts and terminology. The authors discuss trends in hardware, methods, and algorithms; parallel computer architectures and the overall system view of a parallel computer; message-passing; parallelization via multi-threading; measures for predicting and assessing the performance of parallel algorithms; and fundamental issues of designing and implementing parallel programs. The second part contains detailed discussions and performance analyses of parallel algorithms for a number of important and widely used quantum chemistry procedures and methods. The book presents schemes for the parallel computation of two-electron integrals, details the Hartree–Fock procedure, considers the parallel computation of second-order Møller–Plesset energies, and examines the difficulties of parallelizing local correlation methods. Through a solid assessment of parallel computing hardware issues, parallel programming practices, and implementation of key methods, this invaluable book enables readers to develop efficient quantum chemistry software capable of utilizing large-scale parallel computers.

Computer architecture deals with the physical configuration, logical structure, formats, protocols, and operational sequences for processing data, controlling the configuration, and controlling the operations over a computer. It also encompasses word lengths, instruction codes, and the interrelationships among the main parts of a computer or group of computers. This two-volume set offers a comprehensive coverage of the field of computer organization and architecture.

The end of dramatic exponential growth in single-processor performance marks the end of the dominance of the single microprocessor in computing. The era of sequential computing must give way to a new era in which parallelism is at the forefront. Although important scientific and engineering challenges lie ahead, this is an opportune time for innovation in programming systems and computing architectures. We have already begun to see diversity in computer designs to optimize for such considerations as power and throughput. The next generation of discoveries is likely to require advances at both the hardware and software levels of computing systems. There is no guarantee that we can

make parallel computing as common and easy to use as yesterday's sequential single-processor computer systems, but unless we aggressively pursue efforts suggested by the recommendations in this book, it will be "game over" for growth in computing performance. If parallel programming and related software efforts fail to become widespread, the development of exciting new applications that drive the computer industry will stall; if such innovation stalls, many other parts of the economy will follow suit. The Future of Computing Performance describes the factors that have led to the future limitations on growth for single processors that are based on complementary metal oxide semiconductor (CMOS) technology. It explores challenges inherent in parallel computing and architecture, including ever-increasing power consumption and the escalated requirements for heat dissipation. The book delineates a research, practice, and education agenda to help overcome these challenges. The Future of Computing Performance will guide researchers, manufacturers, and information technology professionals in the right direction for sustainable growth in computer performance, so that we may all enjoy the next level of benefits to society.

Parallel computer architectures are now going to real applications! This fact is demonstrated by the large number of application areas covered in this book (see section on applications of parallel computer architectures). The applications range from image analysis to quantum mechanics and data bases. Still, the use of parallel architectures poses serious problems and requires the development of new techniques and tools. This book is a collection of best papers presented at the first workshop on two major research activities at the Universitiit Erlangen-Niirnberg and Technis che Universitiit Miinchen. At both universities, more than 100 researchers are working in the field of multiprocessor systems and network configurations and methods and tools for parallel systems. Indeed, the German Science Founda tion (Deutsche Forschungsgemeinschaft) has been sponsoring the projects under grant numbers SFB 182 and SFB 342. Research grants in the form of a Sonder forschungsbereich are given to selected German Universities in portions of three years following a thoroughful reviewing process. The overall duration of such a research grant is restricted to 12 years. The initiative at Erlangen-Niirnberg was started in 1987 and has been headed since this time by Prof. Dr. H. Wedekind. Work at TU-Miinchen began in 1990, head of this initiative is Prof. Dr. A. Bode. The authors of this book are grateful to the Deutsche Forschungsgemeinschaft for its continuing support in the field of research on parallel processing. The first section of the book is devoted to hardware aspects of parallel systems.

Innovations in hardware architecture, like hyper-threading or multicore processors, mean that parallel computing resources are available for inexpensive desktop computers. In only a few years, many standard software products will be based on concepts of parallel programming implemented on such hardware, and the range of applications will be much broader than that of scientific computing, up to now the main application area for parallel computing. Rauber and Runger take up these recent developments in processor architecture by giving detailed descriptions of parallel programming techniques that are necessary for developing efficient programs for multicore processors as well as for parallel cluster systems and supercomputers. Their book is structured in three main parts, covering all areas of parallel computing: the architecture of parallel systems, parallel programming models and environments, and the implementation of efficient application algorithms. The emphasis lies on parallel programming techniques needed for different architectures. The main goal of the book is to present parallel programming techniques that can be used in many situations for many application areas and which enable the reader to develop correct and efficient parallel programs. Many examples and exercises are provided to show how to apply the techniques. The book can be used as both a textbook for students and a reference book for professionals. The presented material has been used for courses in parallel programming at different universities for many years.

A clear illustration of how parallel computers can be successfully applied to large-scale scientific computations. This book demonstrates how a variety of applications in physics, biology, mathematics and other sciences were implemented on real parallel computers to produce new scientific results. It investigates issues of fine-grained parallelism relevant for future supercomputers with particular emphasis on hypercube architecture. The authors describe how they used an experimental approach to configure different massively parallel machines, design and implement basic system software, and develop algorithms for frequently used mathematical computations. They also devise performance models, measure the performance characteristics of several computers, and create a high-performance computing facility based exclusively on parallel computers. By addressing all issues involved in scientific problem solving, Parallel Computing Works! provides valuable insight into computational science for large-scale parallel architectures. For those in the sciences, the findings reveal the usefulness of an important experimental tool. Anyone in supercomputing and related computational fields will gain a new perspective on the potential contributions of parallelism. Includes over 30 full-color illustrations.

Motivation It is now possible to build powerful single-processor and multiprocessor systems and use them efficiently for data processing, which has seen an explosive ex pansion in many areas of computer science and engineering. One approach to meeting the performance requirements of the applications has been to utilize the most powerful single-processor system that is available. When such a system does not provide the performance requirements, pipelined and parallel process ing structures can be employed. The concept of parallel processing is a depar ture from sequential processing. In sequential computation one processor is in volved and performs one operation at a time. On the other hand, in parallel computation several processors cooperate to solve a problem, which reduces computing time because several operations can be carried out simultaneously. Using several processors that work together on a given computation illustrates a new paradigm in computer problem solving which is completely different from sequential processing. From the practical point of view, this provides sufficient justification to investigate the concept of parallel processing and related issues, such as parallel algorithms. Parallel processing involves utilizing several factors, such as parallel architectures, parallel algorithms, parallel programming lan guages and performance analysis, which are strongly interrelated. In general, four steps are involved in performing a computational problem in parallel. The first step is to understand the nature of computations in the specific application domain.

Algorithmically Specialized Parallel Computers focuses on the concept and characteristics of an algorithmically specialized computer. This book discusses the algorithmically specialized computers, algorithmic specialization using VLSI, and innovative architectures. The architectures and algorithms for digital signal, speech, and image processing and specialized architectures for numerical computations are also elaborated. Other topics include the model for analyzing generalized inter-processor, pipelined architecture for search tree maintenance, and specialized computer organization for raster graphics display. The data base applications of the FETCH-AND-ADD instruction, distributed parallel architecture for speech understanding, and two parallel formulations of particle-in-cell models are likewise covered in this text. This publication is suitable for students, researchers and professionals concerned with algorithmically specialized computers.

Parallel Computing Architectures and APIs: IoT Big Data Stream Processing commences from the point high-performance uniprocessors were becoming increasingly complex, expensive, and power-hungry. A basic trade-off exists between the use of one or a small number of such complex processors, at one extreme, and a moderate to very large number of simpler processors, at the other. When combined with a high-bandwidth, interprocessor communication facility leads to significant simplification of the design process. However, two major roadblocks prevent the widespread adoption of such moderately to massively parallel architectures: the interprocessor communication bottleneck, and the difficulty and high cost of algorithm/software development. One of the most important reasons for studying parallel computing architectures is to learn how to extract the best performance from parallel systems. Specifically, you must understand its architectures so that you will be able to exploit those architectures during programming via the standardized APIs. This

book would be useful for analysts, designers and developers of high-throughput computing systems essential for big data stream processing emanating from IoT-driven cyber-physical systems (CPS). This pragmatic book: Devolves uniprocessors in terms of a ladder of abstractions to ascertain (say) performance characteristics at a particular level of abstraction Explains limitations of uniprocessor high performance because of Moore's Law Introduces basics of processors, networks and distributed systems Explains characteristics of parallel systems, parallel computing models and parallel algorithms Explains the three primary categorical representatives of parallel computing architectures, namely, shared memory, message passing and stream processing Introduces the three primary categorical representatives of parallel programming APIs, namely, OpenMP, MPI and CUDA Provides an overview of Internet of Things (IoT), wireless sensor networks (WSN), sensor data processing, Big Data and stream processing Provides introduction to 5G communications, Edge and Fog computing Parallel Computing Architectures and APIs: IoT Big Data Stream Processing discusses stream processing that enables the gathering, processing and analysis of high-volume, heterogeneous, continuous Internet of Things (IoT) big data streams, to extract insights and actionable results in real time. Application domains requiring data stream management include military, homeland security, sensor networks, financial applications, network management, web site performance tracking, real-time credit card fraud detection, etc.

A complete source of information on almost all aspects of parallel computing from introduction, to architectures, to programming paradigms, to algorithms, to programming standards. It covers traditional Computer Science algorithms, scientific computing algorithms and data intensive algorithms.

Today all computers, from tablet/desktop computers to super computers, work in parallel. A basic knowledge of the architecture of parallel computers and how to program them, is thus, essential for students of computer science and IT professionals. In its second edition, the book retains the lucidity of the first edition and has added new material to reflect the advances in parallel computers. It is designed as text for the final year undergraduate students of computer science and engineering and information technology. It describes the principles of designing parallel computers and how to program them. This second edition, while retaining the general structure of the earlier book, has added two new chapters, 'Core Level Parallel Processing' and 'Grid and Cloud Computing' based on the emergence of parallel computers on a single silicon chip popularly known as multicore processors and the rapid developments in Cloud Computing. All chapters have been revised and some chapters are re-written to reflect the emergence of multicore processors and the use of MapReduce in processing vast amounts of data. The new edition begins with an introduction to how to solve problems in parallel and describes how parallelism is used in improving the performance of computers. The topics discussed include instruction level parallel processing, architecture of parallel computers, multicore processors, grid and cloud computing, parallel algorithms, parallel programming, compiler transformations, operating systems for parallel computers, and performance evaluation of parallel computers.

Computer Architecture/Software Engineering

Parallel and High Performance Computing offers techniques guaranteed to boost your code's effectiveness. Summary Complex calculations, like training deep learning models or running large-scale simulations, can take an extremely long time. Efficient parallel programming can save hours—or even days—of computing time. Parallel and High Performance Computing shows you how to deliver faster run-times, greater scalability, and increased energy efficiency to your programs by mastering parallel techniques for multicore processor and GPU hardware. About the technology Write fast, powerful, energy efficient programs that scale to tackle huge volumes of data. Using parallel programming, your code spreads data processing tasks across multiple CPUs for radically better performance. With a little help, you can create software that maximizes both speed and efficiency. About the book Parallel and High Performance Computing offers techniques guaranteed to boost your code's effectiveness. You'll learn to evaluate hardware architectures and work with industry standard tools such as OpenMP and MPI. You'll master the data structures and algorithms best suited for high performance computing and learn techniques that save energy on handheld devices. You'll even run a massive tsunami simulation across a bank of GPUs. What's inside Planning a new parallel project Understanding differences in CPU and GPU architecture Addressing underperforming kernels and loops Managing applications with batch scheduling About the reader For experienced programmers proficient with a high-performance computing language like C, C++, or Fortran.

About the author Robert Robey works at Los Alamos National Laboratory and has been active in the field of parallel computing for over 30 years. Yuliana Zamora is currently a PhD student and Siebel Scholar at the University of Chicago, and has lectured on programming modern hardware at numerous national conferences. Table of Contents PART 1 INTRODUCTION TO PARALLEL COMPUTING 1 Why parallel computing? 2 Planning for parallelization 3 Performance limits and profiling 4 Data design and performance models 5 Parallel algorithms and patterns PART 2 CPU: THE PARALLEL WORKHORSE 6 Vectorization: FLOPs for free 7 OpenMP that performs 8 MPI: The parallel backbone PART 3 GPUS: BUILT TO ACCELERATE 9 GPU architectures and concepts 10 GPU programming model 11 Directive-based GPU programming 12 GPU languages: Getting down to basics 13 GPU profiling and tools PART 4 HIGH PERFORMANCE COMPUTING ECOSYSTEMS 14 Affinity: Truce with the kernel 15 Batch schedulers: Bringing order to chaos 16 File operations for a parallel world 17 Tools and resources for better code

"Presents the fundamentals of hardware technologies, assembly language, computer arithmetic, pipelining, memory hierarchies and I/O"--

Parallel Computer Architecture A Hardware/software Approach Gulf Professional Publishing

Teaching fundamental design concepts and the challenges of emerging technology, this textbook prepares students for a career designing the computer systems of the future. In-depth coverage of complexity, power, reliability and performance, coupled with treatment of parallelism at all levels, including ILP and TLP, provides the state-of-the-art training that students need. The whole gamut of parallel architecture design options is explained, from core microarchitecture to chip

multiprocessors to large-scale multiprocessor systems. All the chapters are self-contained, yet concise enough that the material can be taught in a single semester, making it perfect for use in senior undergraduate and graduate computer architecture courses. The book is also teeming with practical examples to aid the learning process, showing concrete applications of definitions. With simple models and codes used throughout, all material is made open to a broad range of computer engineering/science students with only a basic knowledge of hardware and software.

The most exciting development in parallel computer architecture is the convergence of traditionally disparate approaches on a common machine structure. This book explains the forces behind this convergence of shared-memory, message-passing, data parallel, and data-driven computing architectures. It then examines the design issues that are critical to all parallel architecture across the full range of modern design, covering data access, communication performance, coordination of cooperative work, and correct implementation of useful semantics. It not only describes the hardware and software techniques for addressing each of these issues but also explores how these techniques interact in the same system. Examining architecture from an application-driven perspective, it provides comprehensive discussions of parallel programming for high performance and of workload-driven evaluation, based on understanding hardware-software interactions. synthesizes a decade of research and development for practicing engineers, graduate students, and researchers in parallel computer architecture, system software, and applications development presents in-depth application case studies from computer graphics, computational science and engineering, and data mining to demonstrate sound quantitative evaluation of design trade-offs describes the process of programming for performance, including both the architecture-independent and architecture-dependent aspects, with examples and case-studies illustrates bus-based and network-based parallel systems with case studies of more than a dozen important commercial designs

Intelligent readers who want to build their own embedded computer systems-- installed in everything from cell phones to cars to handheld organizers to refrigerators-- will find this book to be the most in-depth, practical, and up-to-date guide on the market. Designing Embedded Hardware carefully steers between the practical and philosophical aspects, so developers can both create their own devices and gadgets and customize and extend off-the-shelf systems. There are hundreds of books to choose from if you need to learn programming, but only a few are available if you want to learn to create hardware. Designing Embedded Hardware provides software and hardware engineers with no prior experience in embedded systems with the necessary conceptual and design building blocks to understand the architectures of embedded systems. Written to provide the depth of coverage and real-world examples developers need, Designing Embedded Hardware also provides a road-map to the pitfalls and traps to avoid in designing embedded systems. Designing Embedded Hardware covers such essential topics as: The principles of developing computer hardware Core hardware designs Assembly language concepts Parallel I/O Analog-digital conversion Timers (internal and external) UART Serial Peripheral Interface Inter-Integrated Circuit Bus Controller Area Network (CAN) Data Converter Interface (DCI) Low-power operation This invaluable and eminently useful book gives you the practical tools and skills to develop, build, and program your own application-specific computers.

THE CONTEXT OF PARALLEL PROCESSING The field of digital computer architecture has grown explosively in the past two decades. Through a steady stream of experimental research, tool-building efforts, and theoretical studies, the design of an instruction-set architecture, once considered an art, has been transformed into one of the most quantitative branches of computer technology. At the same time, better understanding of various forms of concurrency, from standard pipelining to massive parallelism, and invention of architectural structures to support a reasonably efficient and user-friendly programming model for such systems, has allowed hardware performance to continue its exponential growth. This trend is expected to continue in the near future. This explosive growth, linked with the expectation that performance will continue its exponential rise with each new generation of hardware and that (in stark contrast to software) computer hardware will function correctly as soon as it comes off the assembly line, has its down side. It has led to unprecedented hardware complexity and almost intolerable development costs. The challenge facing current and future computer designers is to institute simplicity where we now have complexity; to use fundamental theories being developed in this area to gain performance and ease-of-use benefits from simpler circuits; to understand the interplay between technological capabilities and limitations, on the one hand, and design decisions based on user and application requirements on the other.

New sequencing technologies have broken many experimental barriers to genome scale sequencing, leading to the extraction of huge quantities of sequence data. This expansion of biological databases established the need for new ways to harness and apply the astounding amount of available genomic information and convert it into substantive biological understanding. A compilation of recent approaches from prominent researchers, *Bioinformatics: High Performance Parallel Computer Architectures* discusses how to take advantage of bioinformatics applications and algorithms on a variety of modern parallel architectures. Two factors continue to drive the increasing use of modern parallel computer architectures to address problems in computational biology and bioinformatics: high-throughput techniques for DNA sequencing and gene expression analysis—which have led to an exponential growth in the amount of digital biological data—and the multi- and many-core revolution within computer architecture. Presenting key information about how to make optimal use of parallel architectures, this book: Describes algorithms and tools including pairwise sequence alignment, multiple sequence alignment, BLAST, motif finding, pattern matching, sequence assembly, hidden Markov models, proteomics, and evolutionary tree reconstruction Addresses GPGPU technology and the associated massively threaded CUDA programming model Reviews FPGA architecture and programming Presents several parallel algorithms for computing alignments on the Cell/BE architecture, including linear-space pairwise alignment, syntenic alignment, and spliced alignment Assesses underlying concepts and advances in orchestrating the phylogenetic likelihood function on parallel computer architectures (ranging from FPGAs upto the IBM BlueGene/L supercomputer) Covers several effective techniques to fully exploit the computing capability of many-core CUDA-enabled GPUs to accelerate protein sequence database searching, multiple sequence alignment, and motif finding Explains a parallel CUDA-based method for correcting sequencing base-pair errors in HTSR data Because the amount of publicly available sequence data is growing faster than single processor core performance speed, modern bioinformatics tools need to take advantage of parallel computer architectures. Now that the era of the many-core processor has begun, it is expected that future mainstream processors will be parallel systems. Beneficial to anyone actively involved in research and applications, this book helps you to get the most out of these tools and create optimal HPC solutions for bioinformatics.

[Copyright: 5973ccfec26a5c2e116a3f568f58e5d3](https://www.pdfdrive.com/parallel-computer-architecture-a-hardware-software-approach-pdf/ebook/5973ccfec26a5c2e116a3f568f58e5d3)