PARALLEL PROGRAMMING

TECHNIQUES AND APPLICATIONS USING NETWORKED WORKSTATIONS AND PARALLEL COMPUTERS

2nd Edition

BARRY WILKINSON
University of North Carolina at Charlotte
Western Carolina University

MICHAEL ALLEN
University of North Carolina at Charlotte

PEARSON
Prentice Hall

Upper Saddle River, NJ 07458

Library of Congress CataIoging-in-Publication Data

CIP DATA AVAILABLE.

Vice President and Editorial Director, ECS: *Marcia Horton*
Executive Editor: *Kate Hargett*
Vice President and Director of Production and Manufacturing, ESM: *David W. Riccardi*
Executive Managing Editor: *Vince O'Brien*
Managing Editor: *Camille Trentacoste*
Production Editor: *John Keegan*
Director of Creative Services: *Paul Belfanti*
Art Director: *Jayne Conte*
Cover Designer: *Kiwi Design*
Managing Editor, AV Management and Production: *Patricia Burns*
Art Editor: *Gregory Dulles*
Manufacturing Manager: *Trudy Pisciotti*
Manufacturing Buyer: *Lisa McDowell*
Marketing Manager: *Pamela Hersperger*

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN: 0-13-140563-2

Pearson Education Ltd., *London*
Pearson Education Australia Pty. Ltd., *Sydney*
Pearson Education Singapore, Pte. Ltd.
Pearson Education North Asia Ltd., *Hong Kong*
Pearson Education Canada, Inc., *Toronto*
Pearson Educación de Mexico, S.A. de C.V.
Pearson Education—Japan, *Tokyo*
Pearson Education Malaysia, Pte. Ltd.
Pearson Education, Inc., *Upper Saddle River, New Jersey*

To my wife, Wendy,
and my daughter, Johanna

Barry Wilkinson

To my wife, Bonnie

Michael Allen

Preface

The purpose of this text is to introduce parallel programming techniques. Parallel programming is programming multiple computers, or computers with multiple internal processors, to solve a problem at a greater computational speed than is possible with a single computer. It also offers the opportunity to tackle larger problems, that is, problems with more computational steps or larger memory requirements, the latter because multiple computers and multiprocessor systems often have more total memory than a single computer. In this text, we concentrate upon the use of multiple computers that communicate with one another by sending messages; hence the term *message-passing* parallel programming. The computers we use can be different types (PC, SUN, SGI, etc.) but must be interconnected, and a software environment must be present for message passing between computers. Suitable computers (either already in a network or capable of being interconnected) are very widely available as the basic computing platform for students, so that it is usually not necessary to acquire a specially designed multiprocessor system. Several software tools are available for message-passing parallel programming, notably several implementations of MPI, which are all freely available. Such software can also be used on specially designed multiprocessor systems should these systems be available for use. So far as practicable, we discuss techniques and applications in a system-independent fashion.

Second Edition. Since the publication of the first edition of this book, the use of interconnected computers as a high-performance computing platform has become widespread. The term "cluster computing" has come to be used to describe this type of computing. Often the computers used in a cluster are "commodity" computers, that is, low-cost personal computers as used in the home and office. Although the focus of this text, using multiple computers and processors for high-performance computing, has not been changed, we have revised our introductory chapter, Chapter 1, to take into account the move towards commodity clusters and away from specially designed, self-contained, multiprocessors. In the first edition, we described both PVM and MPI and provided an appendix for each. However, only one would normally be used in the classroom. In the second edition, we have deleted specific details of PVM from the text because MPI is now a widely adopted standard and provides for much more powerful mechanisms. PVM can still be used if one wishes, and we still provide support for it on our home page.

Message-passing programming has some disadvantages, notably the need for the programmer to specify explicitly where and when the message passing should occur in the program and what to send. Data has to be sent to those computers that require the data through relatively slow messages. Some have compared this type of programming to assembly language programming, that is, programming using the internal language of the computer, a very low-level and tedious way of programming which is not done except under very specific circumstances. An alternative programming model is the shared memory model. In the first edition, shared memory programming was covered for computers with multiple internal processors and a common shared memory. Such shared memory multiprocessors have now become cost-effective and common, especially dual- and quad-processor systems. Thread programming was described using Pthreads. Shared memory programming remains in the second edition and with significant new material added including performance aspects of shared memory programming and a section on OpenMP, a thread-based standard for shared memory programming at a higher level than Pthreads. Any broad-ranging course on practical parallel programming would include shared memory programming, and having some experience with OpenMP is very desirable. A new appendix is added on OpenMP. OpenMP compilers are available at low cost to educational institutions.

With the focus of using clusters, a major new chapter has been added on shared memory programming on clusters. The shared memory model can be employed on a cluster with appropriate distributed shared memory (DSM) software. Distributed shared memory programming attempts to obtain the advantages of the scalability of clusters and the elegance of shared memory. Software is freely available to provide the DSM environment, and we shall also show that students can write their own DSM systems (we have had several done so). We should point out that there are performance issues with DSM. The performance of software DSM cannot be expected to be as good as true shared memory programming on a shared memory multiprocessor. But a large, scalable shared memory multiprocessor is much more expensive than a commodity cluster.

Other changes made for the second edition are related to programming on clusters. New material is added in Chapter 6 on partially synchronous computations, which are particularly important in clusters where synchronization is expensive in time and should be avoided. We have revised and added to Chapter 10 on sorting to include other sorting algorithms for clusters. We have added to the analysis of the algorithms in the first part of the book to include the computation/communication ratio because this is important to message-passing computing. Extra problems have been added. The appendix on parallel computational models has been removed to maintain a reasonable page count.

The first edition of the text was described as course text primarily for an undergraduate-level parallel programming course. However, we found that some institutions also used the text as a graduate-level course textbook. We have also used the material for both senior undergraduate-level and graduate-level courses, and it is suitable for beginning graduate-level courses. For a graduate-level course, more advanced materials, for example, DSM implementation and fast Fourier transforms, would be covered and more demanding programming projects chosen.

Structure of Materials. As with the first edition, the text is divided into two parts. Part I now consists of Chapters 1 to 9, and Part II now consists of Chapters 10 to 13. In Part I, the basic techniques of parallel programming are developed. In Chapter 1, the concept of parallel computers is now described with more emphasis on clusters. Chapter 2 describes message-passing routines in general and particular software (MPI). Evaluating the performance of message-passing programs, both theoretically and in practice, is discussed. Chapter 3 describes the ideal problem for making parallel the embarrassingly parallel computation where the problem can be divided into independent parts. In fact, important applications can be parallelized in this fashion. Chapters 4, 5, 6, and 7 describe various programming strategies (partitioning and divide and conquer, pipelining, synchronous computations, asynchronous computations, and load balancing). These chapters of Part I cover all the essential aspects of parallel programming with the emphasis on message-passing and using simple problems to demonstrate techniques. The techniques themselves, however, can be applied to a wide range of problems. Sample code is usually given first as sequential code and then as parallel pseudocode. Often, the underlying algorithm is already parallel in nature and the sequential version has "unnaturally" serialized it using loops. Of course, some algorithms have to be reformulated for efficient parallel solution, and this reformulation may not be immediately apparent. Chapter 8 describes shared memory programming and includes Pthreads, an IEEE standard system that is widely available, and OpenMP. There is also a significant new section on timing and performance issues. The new chapter on distributed shared memory programming has been placed after the shared memory chapter to complete Part I, and the subsequent chapters have been renumbered.

Many parallel computing problems have specially developed algorithms, and in Part II problem-specific algorithms are studied in both non-numeric and numeric domains. For Part II, some mathematical concepts are needed, such as matrices. Topics covered in Part II include sorting (Chapter 10), numerical algorithms, matrix multiplication, linear equations, partial differential equations (Chapter 11), image processing (Chapter 12), and searching and optimization (Chapter 13). Image processing is particularly suitable for parallelization and is included as an interesting application with significant potential for projects. The fast Fourier transform is discussed in the context of image processing. This important transform is also used in many other areas, including signal processing and voice recognition.

A large selection of "real-life" problems drawn from practical situations is presented at the end of each chapter. These problems require no specialized mathematical knowledge and are a unique aspect of this text. They develop skills in the use of parallel programming techniques rather than simply teaching how to solve specific problems, such as sorting numbers or multiplying matrices.

Prerequisites. The prerequisite for studying Part I is a knowledge of sequential programming, as may be learned from using the C language. The parallel pseudocode in the text uses C-like assignment statements and control flow statements. However, students with only a knowledge of Java will have no difficulty in understanding the pseudocode, because syntax of the statements is similar to that of Java. Part I can be studied immediately after basic sequential programming has been mastered. Many assignments here can be attempted without specialized mathematical knowledge. If MPI is used for the assignments, programs are usually written in C or C++ calling MPI message-passing library routines. The descriptions of the specific library calls needed are given in Appendix A. It is possible to use Java, although students with only a knowledge of Java should not have any difficulty in writing their assignments in C/C++.

In Part II, the sorting chapter assumes that the student has covered sequential sorting in a data structure or sequential programming course. The numerical algorithms chapter requires the mathematical background that would be expected of senior computer science or engineering undergraduates.

Course Structure. The instructor has some flexibility in the presentation of the materials. Not everything need be covered. In fact, it is usually not possible to cover the whole book in a single semester. A selection of topics from Part I would be suitable as an addition to a normal sequential programming class. We have introduced our first-year students to parallel programming in this way. In that context, the text is a supplement to a sequential programming course text. All of Part I and selected parts of Part II together are suitable as a more advanced undergraduate or beginning graduate-level parallel programming/computing course, and we use the text in that manner.

Home Page. A Web site has been developed for this book as an aid to students and instructors. It can be found at www.cs.uncc.edu/par_prog. Included at this site are extensive Web pages to help students learn how to compile and run parallel programs. Sample programs are provided for a simple initial assignment to check the software environment. The Web site has been completely redesigned during the preparation of the second edition to include step-by-step instructions for students using navigation buttons. Details of DSM programming are also provided. The new Instructor's Manual is available to instructors, and gives MPI solutions. The original solutions manual gave PVM solutions and is still available. The solutions manuals are available electronically from the authors. A very extensive set of slides is available from the home page.

Acknowledgments. The first edition of this text was the direct outcome of a National Science Foundation grant awarded to the authors at the University of North Carolina at Charlotte to introduce parallel programming in the first college year.[1] Without the support of the late Dr. M. Mulder, program director at the National Science Foundation, we would not have been able to pursue the ideas presented in the text. A number of graduate students worked on the original project. Mr. Uday Kamath produced the original solutions manual.

We should like to record our thanks to James Robinson, the departmental system administrator who established our local workstation cluster, without which we would not have been able to conduct the work. We should also like to thank the many students at UNC Charlotte who took our classes and helped us refine the material over many years. This

[1]National Science Foundation grant "Introducing parallel programming techniques into the freshman curricula," ref. DUE 9554975.

included “teleclasses” in which the materials for the first edition were classroom tested in a unique setting. The teleclasses were broadcast to several North Carolina universities, including UNC Asheville, UNC Greensboro, UNC Wilmington, and North Carolina State University, in addition to UNC Charlotte. Professor Mladen Vouk of North Carolina State University, apart from presenting an expert guest lecture for us, set up an impressive Web page that included “real audio” of our lectures and “automatically turning” slides. (These lectures can be viewed from a link from our home page.) Professor John Board of Duke University and Professor Jan Prins of UNC Chapel Hill also kindly made guest-expert presentations to classes. A parallel programming course based upon the material in this text was also given at the Universidad Nacional de San Luis in Argentina by kind invitation of Professor Raul Gallard.

The National Science Foundation has continued to support our work on cluster computing, and this helped us develop the second edition. A National Science Foundation grant was awarded to us to develop distributed shared memory tools and educational materials.[2] Chapter 9, on distributed shared memory programming, describes the work. Subsequently, the National Science Foundation awarded us a grant to conduct a three-day workshop at UNC Charlotte in July 2001 on teaching cluster computing,[3] which enabled us to further refine our materials for this book. We wish to record our appreciation to Dr. Andrew Bernat, program director at the National Science Foundation, for his continuing support. He suggested the cluster computing workshop at Charlotte. This workshop was attended by 18 faculty from around the United States. It led to another three-day workshop on teaching cluster computing at Gujarat University, Ahmedabad, India, in December 2001, this time by invitation of the IEEE Task Force on Cluster Computing (TFCC), in association with the IEEE Computer Society, India. The workshop was attended by about 40 faculty. We are also deeply in the debt to several people involved in the workshop, and especially to Mr. Rajkumar Buyya, chairman of the IEEE Computer Society Task Force on Cluster Computing who suggested it. We are also very grateful to Prentice Hall for providing copies of our textbook to free of charge to everyone who attended the workshops.

We have continued to test the materials with student audiences at UNC Charlotte and elsewhere (including the University of Massachusetts, Boston, while on leave of absence). A number of UNC-Charlotte students worked with us on projects during the development of the second edition. The new Web page for this edition was developed by Omar Lahbabi and further refined by Sari Ansari, both undergraduate students. The solutions manual in MPI was done by Thad Drum and Gabriel Medin, also undergraduate students at UNC-Charlotte.

We would like to express our continuing appreciation to Petra Recter, senior acquisitions editor at Prentice Hall, who supported us throughout the development of the second edition. Reviewers provided us with very helpful advice, especially one anonymous reviewer whose strong views made us revisit many aspects of this book, thereby definitely improving the material.

Finally, we wish to thank the many people who contacted us about the first edition, providing us with corrections and suggestions. We maintained an on-line errata list which was useful as the book went through reprints. All the corrections from the first edition have

[2]National Science Foundation grant “Parallel Programming on Workstation Clusters,” ref. DUE 995030.

[3]National Science Foundation grant supplement for a cluster computing workshop, ref. DUE 0119508.

been incorporated into the second edition. An on-line errata list will be maintained again for the second edition with a link from the home page. We always appreciate being contacted with comments or corrections. Please send comments and corrections to us at wilkinson@email.wcu.edu (Barry Wilkinson) or cma@uncc.edu (Michael Allen).

BARRY WILKINSON
Western Carolina University

MICHAEL ALLEN
University of North Carolina, Charlotte

About the Authors

Barry Wilkinson is a full professor in the Department of Computer Science at the University of North Carolina at Charlotte, and also holds a faculty position at Western Carolina University. He previously held faculty positions at Brighton Polytechnic, England (1984–87), the State University of New York, College at New Paltz (1983–84), University College, Cardiff, Wales (1976–83), and the University of Aston, England (1973–76). From 1969 to 1970, he worked on process control computer systems at Ferranti Ltd. He is the author of *Computer Peripherals* (with D. Horrocks, Hodder and Stoughton, 1980, 2nd ed. 1987), *Digital System Design* (Prentice Hall, 1987, 2nd ed. 1992), *Computer Architecture Design and Performance* (Prentice Hall 1991, 2nd ed. 1996), and *The Essence of Digital Design* (Prentice Hall, 1997). In addition to these books, he has published many papers in major computer journals. He received a B.S. degree in electrical engineering (with first-class honors) from the University of Salford in 1969, and M.S. and Ph.D. degrees from the University of Manchester (Department of Computer Science), England, in 1971 and 1974, respectively. He has been a senior member of the IEEE since 1983 and received an IEEE Computer Society Certificate of Appreciation in 2001 for his work on the IEEE Task Force on Cluster Computing (TFCC) education program.

Michael Allen is a full professor in the Department of Computer Science at the University of North Carolina at Charlotte. He previously held faculty positions as an associate and full professor in the Electrical Engineering Department at the University of North Carolina at Charlotte (1974–85), and as an instructor and an assistant professor in the Electrical Engineering Department at the State University of New York at Buffalo (1968–74). From 1985 to 1987, he was on leave from the University of North Carolina at Charlotte while serving as the president and chairman of DataSpan, Inc. Additional industry experience includes electronics design and software systems development for Eastman Kodak, Sylvania Electronics, Bell of Pennsylvania, Wachovia Bank, and numerous other firms. He received B.S. and M.S. degrees in Electrical Engineering from Carnegie Mellon University in 1964 and 1965, respectively, and a Ph.D. from the State University of New York at Buffalo in 1968.

WilkFMff.fm Page xiii Friday, January 23, 2004 10:51 AM

Contents