McGraw Hill Education

大学计算机教育国外著名教材系列（影印版）

PARALLEL PROGRAMMING IN C WITH MPI AND OPENMP

并行程序设计 C、MPI与OpenMP

Michael J. Quinn 著

清华大学出版社

PARALLEL PROGRAMMING IN C WITH MPI AND OPENMP

并行程序设计 C、MPI与OpenMP

内 容 简 介

本书是美国Oregon州立大学的Miachael J. Quinn教授在多年讲授“并行程序设计”课程的基础上编写而成的，主要介绍用C语言，并结合使用MPI和OpenMP进行并行程序设计，内容包括并行体系结构、并行算法设计、消息传递编程、Eratosthenes筛法、Floyd算法、性能分析、矩阵向量乘法、文档分类、蒙特卡洛法、矩阵乘法、线性方程组求解、有限差分方法、排序、快速傅立叶变换、组合搜索、共享存储编程、融合OpenMP和MPI以及5个附录。

本书按授课方式安排章节，通过划分、通信、集聚和映射等四步的并行程序设计方法，来解决各种实际的并行性问题，使读者掌握系统化的并行程序设计方法，开发出高效的并行程序。

本书不仅是一本优秀的并行程序设计教材，对广大的相关专业人员也很有参考价值。

ISBN 7-302-11157-X

9 787302 111573 >

定价:33.00元

McGraw Hill Education

http://www.mheducation.com

McGraw Hill

影印版

PARALLEL PROGRAMMING
IN C WITH MPI AND OPENMP

并行程序设计 C、MPI与OpenMP

清华大

大学计算机教育国外著名教材系列(影印版)

Parallel Programming in C with MPI and OpenMP

并行程序设计
C、MPI 与 OpenMP

Michael J. Quinn

清华大学出版社
北京

Michael J. Quinn
Parallel Programming in C with MPI and OpenMP
EISBN: 0-07-282256-2

北京市版权局著作权合同登记号　图字:01-2005-0654 号

图书在版编目(CIP)数据

并行程序设计:C、MPI 与 OpenMP 英文/(美)奎因(Quinn, M. J.)著. —影印本. —北京:清华大学出版社, 2005. 8
(大学计算机教育国外著名教材系列)
ISBN 7-302-11157-X

Ⅰ. 并…　Ⅱ. 奎…　Ⅲ. 并行程序—程序设计—高等学校—教材—英文　Ⅳ. TP311. 11

中国版本图书馆 CIP 数据核字(2005)第 058944 号

出 版 者: 清华大学出版社　　**地　　址:** 北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编:** 100084
社总机: 010-62770175　　**客户服务:** 010-62776969
印 刷 者: 北京牛山世兴印刷厂
装 订 者: 三河市金元装订厂
发 行 者: 新华书店总店北京发行所
开　　本: 148×210　**印张:** 16.875
版　　次: 2005 年 8 月第 1 版　2005 年 8 月第 1 次印刷
书　　号: ISBN 7-302-11157-X/TP · 7373
印　　数: 1~3000
定　　价: 33.00 元

出 版 说 明

进入 21 世纪，世界各国的经济、科技以及综合国力的竞争将更加激烈。竞争的中心无疑是对人才的竞争。谁拥有大量高素质的人才，谁就能在竞争中取得优势。高等教育，作为培养高素质人才的事业，必然受到高度重视。目前我国高等教育的教材更新较慢，为了加快教材的更新频率，教育部正在大力促进我国高校采用国外原版教材。

清华大学出版社从 1996 年开始，与国外著名出版公司合作，影印出版了“大学计算机教育丛书(影印版)”等一系列引进图书，受到国内读者的欢迎和支持。跨入 21 世纪，我们本着为我国高等教育教材建设服务的初衷，在已有的基础上，进一步扩大选题内容，改变图书开本尺寸，一如既往地请有关专家挑选适用于我国高等本科及研究生计算机教育的国外经典教材或著名教材，组成本套“大学计算机教育国外著名教材系列(影印版)”，以飨读者。深切期盼读者及时将使用本系列教材的效果和意见反馈给我们。更希望国内专家、教授积极向我们推荐国外计算机教育的优秀教材，以利我们把“大学计算机教育国外著名教材系列(影印版)”做得更好，更适合高校师生的需要。

清华大学出版社

PREFACE

This book is a practical introduction to parallel programming in C using the MPI (Message Passing Interface) library and the OpenMP application programming interface. It is targeted to upper-division undergraduate students, beginning graduate students, and computer professionals learning this material on their own. It assumes the reader has a good background in C programming and has had an introductory class in the analysis of algorithms.

Fortran programmers interested in parallel programming can also benefit from this text. While the examples in the book are in C, the underlying concepts of parallel programming with MPI and OpenMP are essentially the same for both C and Fortran programmers.

In the past twenty years I have taught parallel programming to hundreds of undergraduate and graduate students. In the process I have learned a great deal about the sorts of problems people encounter when they begin "thinking in parallel" and writing parallel programs. Students benefit from seeing programs designed and implemented step by step. My philosophy is to introduce new functionality "just in time." As much as possible, every new concept appears in the context of solving a design, implementation, or analysis problem. When you see the symbol

in a page margin, you'll know I'm presenting a key concept.

The first two chapters explain when and why parallel computing began and gives a high-level overview of parallel architectures. Chapter 3 presents Foster's parallel algorithm design methodology and shows how it is used through several case studies. Chapters 4, 5, 6, 8, and 9 demonstrate how to use the design methodology to develop MPI programs that solve a series of progressively more difficult programming problems. The 27 MPI functions presented in these chapters are a robust enough subset to implement parallel programs for a wide variety of applications. These chapters also introduce functions that simplify matrix and vector I/O. The source code for this I/O library appears in Appendix B.

The programs of Chapters 4, 5, 6, and 8 have been benchmarked on a commodity cluster of microprocessors, and these results appear in the text. Because new generations of microprocessors appear much faster than books can be produced, readers will observe that the processors are several generations old. The point of presenting the results is not to amaze the reader with the speed of the computations. Rather, the purpose of the benchmarking is to demonstrate that knowledge of the latency and bandwidth of the interconnection network, combined with information about the performance of a sequential program, are often sufficient to allow reasonably accurate predictions of the performance of a parallel program.

Chapter 7 focuses on four metrics for analyzing and predicting the performance of parallel systems: Amdahl's Law, Gustafson-Barsis' Law, the Karp-Flatt metric, and the isoefficiency metric.

Chapters 10–16 provide additional examples of how to analyze a problem and design a good parallel algorithm to solve it. At this point the development of MPI programs implementing the parallel algorithms is left to the reader. I present Monte Carlo methods and the challenges associated with parallel random number generation. Later chapters present a variety of key algorithms: matrix multiplication, Gaussian elimination, the conjugate gradient method, finite difference methods, sorting, the fast Fourier transform, backtrack search, branch-and-bound search, and alpha-beta search.

Chapters 17 and 18 are an introduction to the new shared-memory programming standard OpenMP. I present the features of OpenMP as needed to convert sequential code segments into parallel ones. I use two case studies to demonstrate the process of transforming MPI programs into hybrid MPI/OpenMP programs that can exhibit higher performance on multiprocessor clusters than programs based solely on MPI.

This book has more than enough material for a one-semester course in parallel programming. While parallel programming is more demanding than typical programming, it is also more rewarding. Even with a teacher's instruction and support, most students are unnerved at the prospect of harnessing multiple processors to perform a single task. However, this fear is transformed into a feeling of genuine accomplishment when they see their debugged programs run much faster than "ordinary" C programs. For this reason, programming assignments should play a central role in the course.

Fortunately, parallel computers are more accessible than ever. If a commercial parallel computer is not available, it is a straightforward task to build a small cluster out of a few PCs, networking equipment, and free software.

Figure P.1 illustrates the precedence relations among the chapters. A solid arrow from A to B indicates chapter B depends heavily upon material presented in chapter A. A dashed arrow from A to B indicates a weak dependence. If you cover the chapters in numerical order, you will satisfy all of these precedences. However, if you would like your students to start programming in C with MPI as quickly as possible, you may wish to skip Chapter 2 or only cover one or two sections of it. If you wish to focus on numerical algorithms, you may wish to skip Chapter 5 and introduce students to the function `MPI_Bcast` in another way. If you would like to start by having your students programming Monte Carlo algorithms, you can jump to Chapter 10 immediately after Chapter 4. If you want to cover OpenMP before MPI, you can jump to Chapter 17 after Chapter 3.

I thank everyone at McGraw-Hill who helped me create this book, especially Betsy Jones, Michelle Flomenhoft, and Kay Brimeyer. Thank you for your sponsorship, encouragement, and assistance. I also appreciate the help provided by Maggie Murphy and the rest of the compositors at Interactive Composition Corporation.

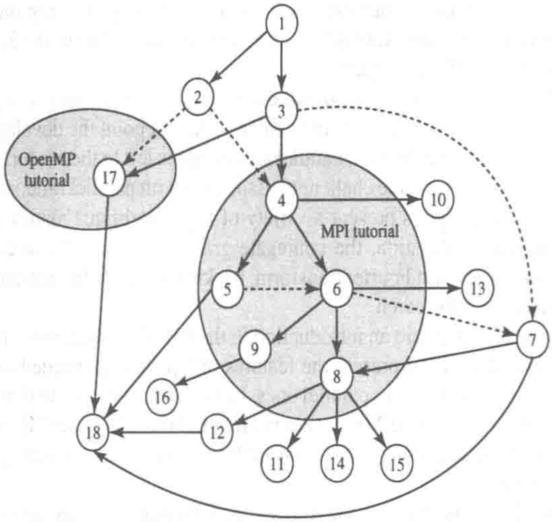

Figure P.1 Dependences among the chapters. A solid arrow indicates a strong dependence; a dashed arrow indicates a weak dependence.

I am indebted to the reviewers who carefully read the manuscript, correcting errors, pointing out weak spots, and suggesting additional topics. My thanks to: A. P. W. Bohm, Colorado State University; Thomas Cormen, Dartmouth College; Narsingh Deo, University of Central Florida; Philip J. Hatcher, University of New Hampshire; Nickolas S. Jovanovic, University of Arkansas at Little Rock; Dinesh Mehta, Colorado School of Mines; Zina Ben Miled, Indiana University-Purdue University, Indianapolis; Paul E. Plassman, Pennsylvania State University; Quinn O. Snell, Brigham Young University; Ashok Srinivasan, Florida State University; Xian-He Sun, Illinois Institute of Technology; Virgil Wallentine, Kansas State University; Bob Weems, University of Texas at Arlington; Kay Zemoudel, California State University-San Bernardino; and Jun Zhang, University of Kentucky.

Many people at Oregon State University also lent me a hand. Rubin Landau and Henri Jansen helped me understand Monte Carlo algorithms and the detailed balance condition, respectively. Students Charles Sauerbier and Bernd Michael Kelm suggested questions that made their way into the text. Tim Budd showed me how to incorporate PostScript figures into LaTeX documents. Jalal Haddad provided technical support. Thank you for your help!

Finally, I am grateful to my wife, Victoria, for encouraging me to get back into textbook writing. Thanks for the inspiring Christmas present: *Chicken Soup for the Writer's Soul: Stories to Open the Heart and Rekindle the Spirit of Writers.*

Michael J. Quinn
Corvallis, Oregon

CONTENTS

CHAPTER **3**
Parallel Algorithm Design 63

CHAPTER **4**
Message-Passing Programming 93

CHAPTER **5**
The Sieve of Eratosthenes 115

CHAPTER **11**

Matrix Multiplication 273

CHAPTER **12**

Solving Linear Systems 290

CHAPTER **13**

Finite Difference Methods 318

CHAPTER **14**

Sorting 338