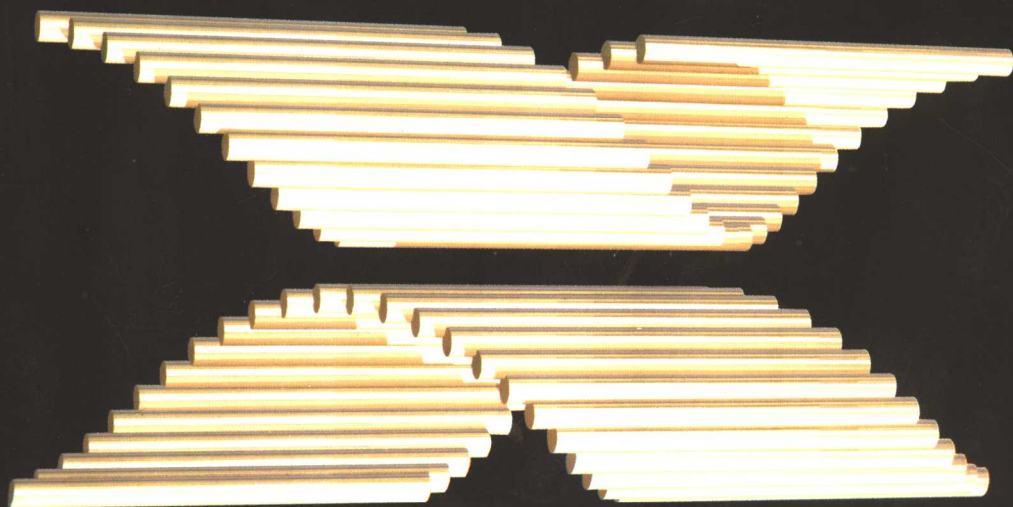


MPI与OpenMP 并行程序设计

C语言版

Michael J. Quinn 著
陈文光 武永卫 等译



PARALLEL PROGRAMMING

in C with MPI and OpenMP

世界著名计算机教材精选

MPI与OpenMP并行程序设计

(C语言版)

Michael J. Quinn 著

陈文光 武永卫 等译

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

Copyright © 2004 by The McGraw-Hill Companies, Inc.

Original language published by The McGraw-Hill Companies, Inc. All Rights reserved. No part of this publication may be reproduced or distributed by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

Simplified Chinese translation edition is published and distributed exclusively by Tsinghua University Press under the authorization by McGraw-Hill Education (Asia) Co., within the territory of the People's Republic of China only, excluding Hong Kong, Macao SAR and Taiwan. Unauthorized export of this edition is a violation of the Copyright Act. Violation of this Law is subject to Civil and Criminal Penalties.

本书中文简体字翻译版由美国麦格劳-希尔教育出版(亚洲)公司授权清华大学出版社在中华人民共和国境内(不包括中国香港、澳门特别行政区和中国台湾)独家出版发行。未经许可之出口,视为违反著作权法,将受法律之制裁。未经出版者预先书面许可,不得以任何方式复制或抄袭本书的任何部分。

北京市版权局著作权合同登记号 图字 01-2003-7768 号

版权所有,翻印必究。举报电话:010-62782989 13901104297 13801310933

本书封面贴有 McGraw-Hill 公司防伪标签,无标签者不得销售。

图书在版编目(CIP)数据

MPI 与 OpenMP 并行程序设计: C 语言版/奎因(Quinn, M. J.)著;陈文光,武永卫等译. —北京:清华大学出版社, 2004.10

(世界著名计算机教材精选)

书名原文: Parallel Programming in C with MPI and OpenMP

ISBN 7-302-09555-8

I. M… II. ①奎… ②陈… ③武… III. 并行程序-程序设计-教材 IV. TP311.11

中国版本图书馆 CIP 数据核字(2004)第 095663 号

出版者: 清华大学出版社

<http://www.tup.com.cn>

社总机: 010-62770175

地址: 北京清华大学学研大厦

邮编: 100084

客户服务: 010-62776969

责任编辑: 龙啟铭

印刷者: 世界知识印刷厂

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: 185×260 印张: 27.75 字数: 687 千字

版次: 2004 年 10 月第 1 版 2004 年 10 月第 1 次印刷

书号: ISBN 7-302-09555-8/TP·6645

印数: 1~3000

定价: 51.00 元

本书如存在文字不清、漏印以及缺页、倒页、脱页等印装质量问题,请与清华大学出版社出版部联系调换。联系电话: (010) 62770175-3103 或 (010) 62795704

前 言

本书是用 C 语言进行 MPI 和 OpenMP 并行编程的实用教程，适用于大学高年级本科生、研究生以及利用本书进行自学的计算机专业人员。读者需要有很好的 C 语言编程经验，并学习过基础的算法分析课程。

对并行程序设计感兴趣的 Fortran 程序员也可以从本书中受益。尽管本书中的例子都是用 C 写的，但是使用 MPI 和 OpenMP 进行并行程序设计的基本概念对于 C 和 Fortran 来说是基本相同的。

在过去 20 年中，我为数以百计的本科生和研究生讲授了并行程序设计。在这个过程中，我逐步了解了人们在开始“用并行方式思考”和编写并行程序时所遇到的各种问题。逐步设计和实现的程序更容易让学生们受益。因此我的哲学是仅在需要时才引入新的功能，尽可能地在解决设计、实现和分析中的具体问题时引入新概念。

本书的前两章解释了并行计算的起源并对并行体系结构进行了综述。第 3 章介绍了 Foster 的并行算法设计方法论和几个应用该方法的实例。第 4、5、6、8 和 9 章介绍了如何使用该设计方法，为一系列从易到难的问题开发 MPI 程序。这些章节中用到的 27 个 MPI 函数是 MPI 函数库的一个子集，但已经足够为很多类型的实际应用设计并行程序。这些章节还介绍了能够简化矩阵和向量 I/O 的函数，附录 B 中给出了该 I/O 库的源代码。

第 4、5、6 和 8 章的程序在一个集群系统上进行了性能测试，并在书中给出了测试结果。由于新的处理器比本书中所用到的要快得多，读者可能会发现本书中用到的处理器已经落后了好几代。但是在书中给出测试结果并不是要让读者对计算速度感到吃惊，而是为了表明将串行程序性能、互连网络的通信和延迟等信息集成起来，可以相当准确地预测并行程序的性能。

第 7 章主要介绍了分析和预测并行系统性能所用到的 4 种度量：Amdahl 定律、Gustafson-Barsis 定律、Karp-Flatt 度量和等效度量。

第 10~16 章提供了更多的例子，表明如何分析问题并设计好的并行算法来解决问题。使用 MPI 来实现并行算法的任务则留给了读者。我介绍了蒙特卡洛法和并行随机数生成的相关问题。后面几章介绍了若干关键算法：矩阵乘法、高斯消去法、共轭梯度法、有限元方法、排序、FFT、回溯搜索、分支定界搜索以及 Alpha-Beta 搜索。

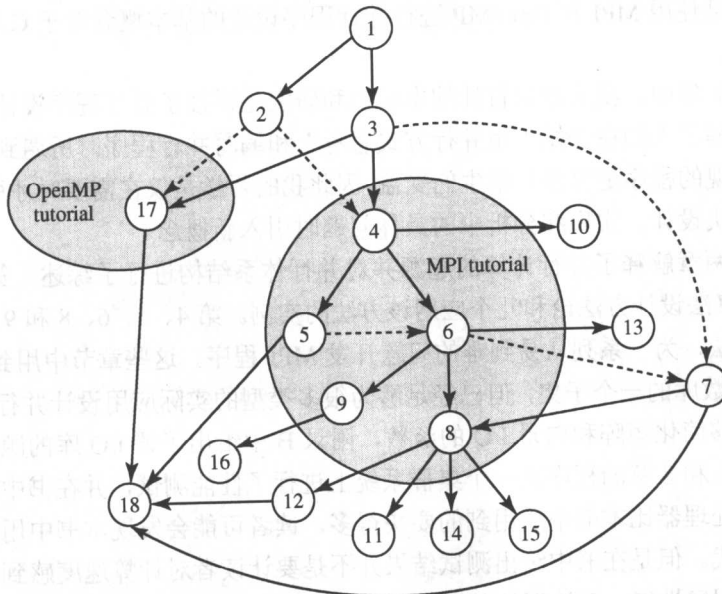
第 17、18 章介绍了新的共享内存编程标准 OpenMP。介绍了将串行程序段转化为并行程序段所需的 OpenMP 功能，并用两个实例讲解了如何将 MPI 程序转化为 MPI/OpenMP 混合程序。在多处理器集群系统上，混合程序比 MPI 程序能够得到更好的性能。

本书包含了超过一学期并行程序设计课程所需的内容。尽管并行程序设计比传统程序设计要困难得多，但也会带来更多的回报。即使在老师的指导与帮助下，大多数学生对于在多个处理器上执行单个任务仍然会觉得有些害怕。但是，当他们那看到调试过的程序比“普通”C 程序要快得多的时候，这种害怕就转换成了真正的成就感。因此，编程练习应

该是本课程的中心内容。

幸运的是，并行计算机比以前更容易获得了。如果无法使用商用并行计算机，使用几台 PC 机、网络设备和自由软件搭建一个小型的集群系统是一件很容易的事情。

图前.1 说明了本书章节的阅读顺序。图中实线箭头代表强依赖关系，虚线箭头代表弱依赖关系。按照章节顺序阅读本书将满足所有的章节依赖关系。但是，如果你希望让学生尽快开始用 C 语言编写 MPI 程序，那么你可能希望跳过第 2 章，或仅讲授该章的 1~2 节内容。如果你希望集中讲授数值算法，那么可以跳过第 5 章，并用其他方式介绍函数 `MPI_Bcast`。如果你希望学生从蒙特卡洛法开始，那么可以在第 4 章后直接跳到第 10 章。如果你希望在 MPI 之前就讲授 OpenMP，可以在第 3 章后直接跳到第 17 章。



图前.1 章节之间的依赖关系：实线箭头代表强依赖关系，虚线箭头代表弱依赖关系

感谢 McGraw-Hill 出版社的每位员工，他们帮助我创造了这本书，特别是 Betsy Jones, Michelle Flomenhoft 和 Kay Brimeyer。感谢他们的赞助、鼓励和帮助。同样感谢 Maggie Murphy 和 Interactive Composition Corporation 公司其他排版者所提供的帮助。

感谢本书的审稿者，他们仔细阅读了我的手稿，修正了其中的错误，指出其中的弱点并建议补充内容。他们是：A. P. W. Bohm, Colorado 州立大学；Thomas Cormen, Dartmouth 学院；Narsingh Deo, Central Florida 大学；Philip J. Hatcher, New Hampshire 大学；Nickolas S. Jovanovic, Arkansas 大学 Little Rock 分校；Dinesh Mehta, Colorado 矿业学校；Zina Ben Miled, Indiana 大学-Purdue 大学；Paul E. Plassman, Pennsylvania 州立大学；Quinn O. Snell, Brigham Young 大学；Ashok Srinivasan, Florida 州立大学；Xian-He Sun, Illinois 理工学院；Virgil Wallentine, Kansas 州立大学；Bob Weems, Texas 大学 Arlington 分校；Kay Zemoudeh, California 州立大学和 Jun Zhang, Kentucky 大学。

Oregon 州立大学的许多同事也给予了我不少帮助：Robin Landaw 和 Henri Jansen 分别帮助我了解了 Monte Carlo 算法和详细的平衡条件；学生 Charles Sauerbier 和 Bernd Michael

Kelm; Tim Budd 教我怎么把 PostScript 图插入到 LaTeX 文档中; Jalal Haddad 提供了技术支持。感谢他们的帮助!

最后,感谢我的妻子 Victoria。她鼓励我返回到教科书的写作中来。感谢那个令人鼓舞的圣诞节礼物:“Chicken Soup for the Writer’s Soul: Stories to Open the Heart and Rekindle the Spirit of Writers”。

译者序

并行处理是解决人类重大挑战问题的关键技术。随着集群技术和 SMP 系统的发展，并行处理在科学研究、工程计算以及商业计算等领域得到了越来越多的应用。并行程序设计一直是并行处理技术中的核心问题，人们也进行了非常多的尝试，提出了多种并行程序开发方法和并行程序设计语言。近年来，MPI 和 OpenMP 逐渐成为了并行程序设计的主流方式。

本书是介绍使用 MPI 和 OpenMP 进行并行程序设计的教材，作者是美国俄勒冈州立大学的 Michael J. Quinn 教授。我们觉得此书具有下面几个特点：

- 作为一本程序设计教材，本书没有简单地罗列所涉及的语句和函数，而是为每个语句和函数的“出场”都精心设计了实际问题，让读者能够在实际的上下文中了解这些语句和函数的目的和作用，以及在实际程序中的使用方式。这无疑将对读者理解相关的概念带来极大的帮助。
- 全书中贯穿使用了 Foster 提出的并行程序设计方法，即划分、通信、聚集和映射四步法。掌握系统化的并行程序设计方法，有助于读者养成良好的习惯，做出正确的设计决策，开发出高效率的并行程序。
- 并行程序的性能是非常重要的，一个低效的并行程序可能还没有串行程序速度快。本书专门介绍了并行程序性能度量模型和性能分析技术，并在多个章节中进行了实例分析，对并行程序的性能问题给予了足够的重视。
- 内容较新。尽管目前已经有较多讲授 MPI（消息传递接口）并行程序设计的教材，但本书不仅包括了 MPI 并行程序设计，还包括了 OpenMP 程序设计的内容。这是目前许多并行程序设计教材所不具备的。本书还介绍了 MPI/OpenMP 混合编程模式，该模式对于现有的 SMP（对称多处理器）集群系统具有重要意义。此外，处理器在未来一段时间内的发展趋势是在一个芯片上集成多个核心，研究 MPI/OpenMP 混合编程模式对采用多内核处理器构造的并行系统的适用性也是一个重要的研究课题。

本书由于篇幅所限，并未在所有有关问题上都进行详细的展开讲解，但每一章最后的参考文献为希望更加深入学习的读者提供了进一步阅读的建议。

因此我们认为，本书是一本很优秀的并行程序设计教材，适合学习并行程序设计的学生和计算机专业人士阅读。我们希望本书中文版的出版，能够将本书介绍给更多的中国读者，并为大家的阅读带来方便。同时，由于本书所涉及的领域相当广泛，译者水平有限，翻译中可能还存在不妥之处，敬请广大读者批评指正。

感谢清华大学计算机系的周立柱教授，将本书介绍给我们。感谢清华大学出版社的龙啟铭编辑，为本书的中文版的出版所付出的耐心和努力。

全书由陈文光、武永卫、陈永健、李建江、薛瑞尼、齐琳等译。清华大学计算机系都志辉进行了审读。

译者

目 录

第 1 章 动机和历史1	2.2.5 超树形网络..... 24
1.1 概述.....1	2.2.6 蝶形网络..... 25
1.2 现代科学方法.....2	2.2.7 超立方体网络..... 26
1.3 超级计算的进化.....3	2.2.8 混洗-交换网络..... 27
1.4 现代并行计算机.....4	2.2.9 小结..... 28
1.4.1 Cosmic Cube 并行计算机.....4	2.3 阵列处理机..... 29
1.4.2 商品化的并行计算机.....5	2.3.1 体系结构与数据并行..... 29
1.4.3 Beowulf 系统.....6	2.3.2 阵列处理机的性能..... 30
1.4.4 先进战略计算计划.....6	2.3.3 处理器互连网络..... 31
1.5 寻找并行性.....7	2.3.4 处理器的启动与阻塞..... 32
1.5.1 数据相关图.....7	2.3.5 其他体系结构特点..... 33
1.5.2 数据并行性.....8	2.3.6 阵列处理机的缺点..... 33
1.5.3 功能并行性.....9	2.4 多处理器..... 33
1.5.4 流水线.....9	2.4.1 集中式多处理器..... 34
1.5.5 计算规模的考虑因素.....11	2.4.2 分布式多处理器..... 35
1.6 数据聚类.....11	2.5 多计算机..... 38
1.7 为并行计算机编程.....13	2.5.1 非对称多计算机..... 39
1.7.1 扩展编译器.....13	2.5.2 对称多计算机..... 40
1.7.2 扩展串行编程语言.....14	2.5.3 怎样的模型对商用集群来说是最佳的..... 41
1.7.3 增加并行编程层.....14	2.5.4 集群与工作站网络之间的差异..... 42
1.7.4 创造一个并行语言.....15	2.6 弗林分类法..... 42
1.7.5 现状.....16	2.6.1 SISD..... 43
1.8 本章小结.....16	2.6.2 SIMD..... 43
1.9 主要术语.....16	2.6.3 MISD..... 43
1.10 参考文献.....17	2.6.4 MIMD..... 45
1.11 练习题.....18	2.7 本章小结..... 45
第 2 章 并行体系结构21	2.8 主要术语..... 46
2.1 概述.....21	2.9 参考文献..... 47
2.2 互连网络.....21	2.10 练习题..... 47
2.2.1 共享介质与开关介质.....22	第 3 章 并行算法设计 50
2.2.2 开关网络的拓扑结构.....22	3.1 概述..... 50
2.2.3 二维网格形网络.....23	
2.2.4 二叉树形网络.....23	

3.2	任务/通道模型	50	4.4.2	MPI_Comm_rank 和 MPI_Comm_size 函数	80
3.3	Foster 的设计方法论	51	4.4.3	MPI_Finalize 函数	81
3.3.1	划分	52	4.4.4	编译 MPI 程序	81
3.3.2	通信	53	4.4.5	运行 MPI 程序	81
3.3.3	聚集	54	4.5	聚合通信简介	83
3.3.4	映射	55		MPI_Reduce 函数	84
3.4	边界值问题	58	4.6	检测并行性能	86
3.4.1	简介	58	4.6.1	MPI_Wtime 和 MPI_Wtick 函数	87
3.4.2	划分	59	4.6.2	MPI_Barrier 函数	87
3.4.3	通信	59	4.7	本章小结	88
3.4.4	聚集与映射	60	4.8	主要术语	89
3.4.5	分析	60	4.9	参考文献	89
3.5	找出最大值	60	4.10	练习题	89
3.5.1	简介	60	第 5 章	Eratosthenes 筛法	93
3.5.2	划分	61	5.1	概述	93
3.5.3	通信	61	5.2	串行算法	93
3.5.4	聚集与映射	64	5.3	并行性的来源	94
3.5.5	分析	65	5.4	数据分解方法	95
3.6	n-body 问题	65	5.4.1	交叉数据分解	95
3.6.1	简介	65	5.4.2	按块数据分解	95
3.6.2	划分	65	5.4.3	用于按块分解的宏	96
3.6.3	通信	66	5.4.4	局部下标还是全局下标	97
3.6.4	聚集与映射	67	5.4.5	块分解的结果	97
3.6.5	分析	67	5.5	开发并行算法	97
3.7	增加数据输入	68		函数 MPI_Bcast	98
3.7.1	简介	68	5.6	并行筛法算法的分析	99
3.7.2	通信	69	5.7	并行程序的说明	99
3.7.3	分析	69	5.8	测试	104
3.8	本章小结	70	5.9	改进	105
3.9	主要术语	70	5.9.1	删除偶数	105
3.10	参考文献	71	5.9.2	消除广播	106
3.11	练习题	71	5.9.3	循环的重新组织	106
			5.9.4	测试	106
第 4 章	消息传递编程	74	5.10	本章小结	108
4.1	概述	74	5.11	主要术语	108
4.2	消息传递模型	74	5.12	参考文献	108
4.3	MPI 接口	76			
4.4	电路可满足性问题	76			
4.4.1	MPI_Init 函数	80			

5.13 练习题	108	8.4 矩阵按行分解	145
第 6 章 Floyd 算法	111	8.4.1 设计与分析	145
6.1 概述	111	8.4.2 复制分块的向量	146
6.2 全点对最短路径问题	111	8.4.3 函数 MPI_Allgatherv	147
6.3 运行时创建数组	112	8.4.4 被复制向量的输入/输出	149
6.4 设计并行算法	113	8.4.5 编写并行程序	149
6.4.1 划分	113	8.4.6 测试	150
6.4.2 通信	114	8.5 矩阵按列分解	151
6.4.3 聚合和映射	115	8.5.1 设计与分析	151
6.4.4 矩阵的输入/输出	116	8.5.2 读取按列分解的矩阵	152
6.5 点对点通信	117	8.5.3 函数 MPI_Scatterv	153
6.5.1 函数 MPI_Send	118	8.5.4 打印输出按列分块矩阵	154
6.5.2 函数 MPI_Recv	119	8.5.5 函数 MPI_Gatherv	154
6.5.3 死锁	120	8.5.6 分发中间结果	155
6.6 并行程序的说明	121	8.5.7 函数 MPI_Alltoallv	156
6.7 分析和测试	123	8.5.8 编写并行程序	156
6.8 本章小结	124	8.5.9 测试	158
6.9 主要术语	125	8.6 棋盘式分解	159
6.10 参考文献	125	8.6.1 设计与分析	159
6.11 练习题	125	8.6.2 创建通信域	162
第 7 章 性能分析	128	8.6.3 函数 MPI_Dims_create	162
7.1 概述	128	8.6.4 函数 MPI_Cart_create	163
7.2 加速比和效率	128	8.6.5 读取棋盘式矩阵	163
7.3 Amdahl 定律	130	8.6.6 函数 MPI_Cart_rank	164
7.3.1 Amdahl 定律的局限	131	8.6.7 函数 MPI_Cart_coords	165
7.3.2 Amdahl 效应	132	8.6.8 函数 MPI_Comm_split	165
7.4 Gustafson-Barsis 定律	132	8.6.9 测试	166
7.5 Karp-Flatt 量度	134	8.7 本章小结	167
7.6 等效指标	136	8.8 主要术语	168
7.7 本章小结	139	8.9 参考文献	168
7.8 主要术语	140	8.10 练习题	169
7.9 参考文献	141	第 9 章 文档分类	173
7.10 练习题	141	9.1 概述	173
第 8 章 矩阵向量乘法	143	9.2 并行算法设计	173
8.1 概述	143	9.2.1 划分与通信	174
8.2 串行算法	143	9.2.2 聚集和映射	174
8.3 数据分解方式	144	9.2.3 管理者/工人模式	174
		9.2.4 管理进程	175

9.2.5	MPI_Abort 函数	176	10.4.3	拒绝法	202
9.2.6	工人进程	177	10.5	应用示例	204
9.2.7	建立一个只有工人的通信域	178	10.5.1	中子输运	204
9.3	非阻塞通信	179	10.5.2	二维板上一个点的温度	206
9.3.1	管理进程的通信	180	10.5.3	二维易辛模型	207
9.3.2	MPI_Irecv 函数	180	10.5.4	房间分配问题	209
9.3.3	MPI_Wait 函数	180	10.5.5	车库停车问题	212
9.3.4	工人的通信	180	10.5.6	交通环路	213
9.3.5	MPI_Isend 函数	181	10.6	本章小结	216
9.3.6	MPI_Probe 函数	181	10.7	主要术语	216
9.3.7	MPI_Get_count 函数	181	10.8	参考文献	217
9.4	文档分类的并行程序	181	10.9	练习题	218
9.5	算法改进	187	第 11 章	矩阵乘法	220
9.5.1	按组分配文档	187	11.1	概述	220
9.5.2	流水线处理	187	11.2	矩阵相乘的串行算法	220
9.5.3	MPI_Testsome 函数	189	11.2.1	基于行的迭代算法	220
9.6	本章小结	189	11.2.2	基于块的递归算法	222
9.7	主要术语	190	11.3	行块分解并行算法	224
9.8	参考文献	190	11.3.1	确定原始任务	224
9.9	练习题	190	11.3.2	聚合	224
			11.3.3	通信和进一步的聚合	225
第 10 章	蒙特卡洛法	193	11.3.4	分析	226
10.1	概述	193	11.4	Cannon 算法	227
10.1.1	为什么蒙特卡洛法能奏效	195	11.4.1	组合	227
10.1.2	蒙特卡洛法与并行计算	196	11.4.2	通信	228
10.2	串行随机数生成器	196	11.4.3	分析	229
10.2.1	线性同余法	197	11.5	本章小结	230
10.2.2	滞后形斐波那契生成器	197	11.6	主要术语	231
10.3	并行随机数产生器	198	11.7	参考文献	231
10.3.1	管理者-工人方法	198	11.8	练习题	231
10.3.2	蛙跳方法	198	第 12 章	线性方程组求解	233
10.3.3	序列分割	199	12.1	概述	233
10.3.4	参数化	199	12.2	基本术语	233
10.4	其他的随机数分布	200	12.3	回代法	234
10.4.1	逆分布累积分布函数变换	200	12.3.1	串行算法	234
10.4.2	Box-Muller 变换	201	12.3.2	面向行的并行算法	236
			12.3.3	面向列的并行算法	236

12.3.4 对比	237	14.1 概述	271
12.4 高斯消去法	237	14.2 快速排序	271
12.4.1 串行算法	237	14.3 并行快速排序算法	272
12.4.2 并行算法	239	14.3.1 排序完毕的定义	273
12.4.3 面向行的算法	240	14.3.2 算法开发	273
12.4.4 面向列的算法	242	14.3.3 分析	273
12.4.5 对比	242	14.4 超级快速排序	274
12.4.6 面向行的流水线算法	243	14.4.1 算法描述	274
12.5 迭代法	244	14.4.2 等效分析	275
12.6 共轭梯度法	247	14.5 规则取样并行排序	277
12.6.1 串行算法	247	14.5.1 算法描述	277
12.6.2 并行算法	249	14.5.2 等效分析	277
12.7 本章小结	250	14.6 本章小结	279
12.8 主要术语	251	14.7 主要术语	280
12.9 参考文献	251	14.8 参考文献	280
12.10 练习题	252	14.9 练习题	280
第 13 章 有限差分方法	254	第 15 章 快速傅立叶变换	283
13.1 概述	254	15.1 概述	283
13.2 偏微分等式	255	15.2 傅立叶分析	283
13.2.1 偏微分方程的分类	255	15.3 离散傅立叶变换	285
13.2.2 差分商	256	15.3.1 离散傅立叶逆变换	286
13.3 弦振荡问题	257	15.3.2 应用示例: 多项式乘法	286
13.3.1 导出方程	257	15.4 快速傅立叶变换	288
13.3.2 串行程序	259	15.5 并行程序设计	291
13.3.3 并行程序设计	260	15.5.1 分割与通信	291
13.3.4 等效分析	261	15.5.2 聚合与映射	292
13.3.5 冗余计算	262	15.5.3 等效分析	292
13.4 稳定状态热量分布问题	263	15.6 本章小结	294
13.4.1 方程的导出	263	15.7 主要术语	294
13.4.2 串行程序导出	264	15.8 参考文献	294
13.4.3 并行程序设计	265	15.9 练习题	294
13.4.4 等效分析	266	第 16 章 组合搜索	296
13.4.5 实现细节	267	16.1 概述	296
13.5 本章小结	267	16.2 回溯搜索	297
13.6 主要术语	268	16.2.1 示例	297
13.7 参考文献	268	16.2.2 时间和空间复杂性	298
13.8 练习题	269	16.3 并行回溯算法	298
第 14 章 排序	271	16.4 分布式终止检测	302

16.5	分支定界法	304	17.6	归约操作	332
16.5.1	示例	304	17.7	性能改善	333
16.5.2	串行算法	306	17.7.1	循环转化	333
16.5.3	分析	308	17.7.2	条件执行循环	334
16.6	并行分支定界法	308	17.7.3	循环调度	335
16.6.1	存储和共享待解的 子问题	308	17.8	更普遍的数据并行	336
16.6.2	效率	309	17.8.1	parallel 编译指导语句	338
16.6.3	停机条件	309	17.8.2	omp_get_thread_num 函数	339
16.7	搜索博弈树	312	17.8.3	omp_get_num_threads 函数	340
16.7.1	最大最小算法	312	17.8.4	编译指导语句 for	340
16.7.2	Alpha-Beta 剪枝	313	17.8.5	single 编译指导语句	342
16.7.3	Alpha-Beta 剪枝法 的改进	315	17.8.6	nowait 子句	342
16.8	并行 Alpha-Beta 搜索	316	17.9	功能并行	343
16.8.1	并行渴望搜索	316	17.9.1	parallel sections 编译 指导语句	343
16.8.2	并行子树估值	316	17.9.2	section 编译指导语句	344
16.8.3	分布式树搜索	317	17.9.3	sections 编译指导语句	344
16.9	本章小结	318	17.10	本章小结	345
16.10	主要术语	319	17.11	主要术语	347
16.11	参考文献	320	17.12	参考文献	347
16.12	练习题	320	17.13	练习题	347
第 17 章	共享存储编程	323	第 18 章	融合 OpenMP 和 MPI	350
17.1	概述	323	18.1	概述	350
17.2	共享存储模型	324	18.2	共轭梯度算法	351
17.3	对 for 循环的并行化	325	18.2.1	MPI 程序	351
17.3.1	parallel for 编译 指导语句	325	18.2.2	函数级程序轮廓刻画	354
17.3.2	omp_get_num_procs 函数	327	18.2.3	对函数 matrix_vector_ product 进行并行化	355
17.3.3	omp_set_num_threads 函数	327	18.2.4	测试程序	355
17.4	声明私有变量	328	18.3	Jacobi 方法	356
17.4.1	private 子句	328	18.3.1	MPI 程序轮廓刻画	356
17.4.2	firstprivate 子句	329	18.3.2	对函数 find_steady_state 并行化	357
17.4.3	lastprivate 子句	330	18.3.3	测试程序	359
17.5	临界区	330	18.4	本章小结	360
	critical 编译指导语句	331	18.5	练习题	360

附录 A MPI 函数	362	C.2.2 导致不准确结果的错误	413
附录 B 工具函数	393	C.2.3 组通信的优点	413
B.1 MyMPI.h 头文件	393	C.3 实用调试策略	413
B.2 MyMPI.c 源文件	394	附录 D 复数回顾	415
附录 C 调试 MPI 程序	412	附录 E OpenMP 函数	418
C.1 概述	412	参考文献	420
C.2 MPI 程序常见错误	412		
C.2.1 死锁错误	412		

第 1 章 动机和历史

Well done is quickly done.

Caesar Augustus

1.1 概 述

你是那些认为运行速度还不够“快”的人吗？现在的计算机工作站比10年前的要快了100倍，但是计算科学家和工程师还需要更快的速度。他们已经对问题进行了大量的简化，但是在现有的计算机上仍然需要数小时、数日甚至数周的时间来完成他们的程序。

更快的计算机可以处理更加复杂的计算问题。假如你可以等待一个晚上来得到计算结果。当你的程序突然快了10倍的时候，过去一些无法忍受的计算任务现在变得可以接受了，过去需要大约一周时间的计算，现在可以用15小时左右来完成了。

当然，你可以等待CPU运行得更快。5年后的CPU将比现在快10倍（根据摩尔定理）。但另一方面，如果你可以等待5年的时间，你对速度的要求肯定没有那么多高！并行计算现在是现在就获得更高性能的有效方法。

什么是并行计算

并行计算就是使用并行计算机来减少解决单个计算问题所需的时间。现在，并行计算被认为是科学家和工程师用来解决各种领域的问题的标准方法，如银河系的演变过程、气候模拟、飞行器设计以及分子动力学等。

什么是并行计算机

并行计算机是支持并行计算的多处理器计算机系统。多计算机（multicomputer）和集中式多处理器（centralized multiprocessors）是两种主要的并行计算机。多计算机是由多台计算机和互连网络组成的并行计算机。不同计算机上的处理器之间通过传递消息来相互通信。

相反，集中式多处理器（也称为对称多处理器系统，SMP 或 symmetrical multiprocessor）是集成的更加紧密的系统。系统中的所有CPU共享全局内存，并通过共享内存支持处理器之间的通信和同步。

在第2章，我们将研究集中式多处理器、多计算机以及其他类型的并行计算机。

什么是并行程序设计

并行程序设计是使用程序设计语言显式地说明计算中不同部分如何在不同处理器上同时执行。在本章的最后，我们将讨论各种并行程序设计语言。

并行程序设计真的有必要吗

人们投入了很多的精力来研究自动并行化技术，目标是能够自动将 Fortran77 或 C 语言程序转换成在拥有大量处理器的并行计算机上可执行的高效并行代码。这是一个非常困难的问题，尽管出现了一些试验性的并行化编译器，但是迄今为止，还没有出现任何成熟的商业产品。因此，我们只能自己编写并行程序。

为什么应该使用 MPI 或 OpenMP 编程

MPI (Message Passing Interface, 消息传递接口) 是消息传递库的标准。几乎所有的并行计算机都支持该标准通信库。如果你希望在集群系统上运行 MPI 的话，也可以使用免费的 MPI 库。如果使用 MPI 开发程序，当你可以使用更新、更快的系统的时候，可以重用该程序而无需重新编写。

并行计算机正越来越多地使用 SMP 系统来构建。在每个 SMP 系统内部，CPU 共享全局地址空间。尽管 MPI 是在不同 SMP 系统之间进行通信的良好方式，但 OpenMP 在描述单个 SMP 节点内部的处理器之间通信上更加有效。在第 18 章，我们将会看到，在实际应用程序中采用 MPI/OpenMP 混合编程，可以获得比仅使用 MPI 编程更好的性能。

通过本书的学习，你可以学到一些关于并行计算机硬件的知识，以及很多并行程序设计策略方面的知识，包括并行算法设计和分析，程序实现和调试，以及测试和优化程序的方法。

1.2 现代科学方法

经典科学是基于观察、理论和实际实验的。通过对现象的观察，总结出假说。科学家发展出理论来解释现象，并设计实验来对理论进行测试。通常，实验的结果会导致科学家对理论进行修正，或是完全放弃。于是，观察又成为中心的步骤。

经典科学的特征是模型和实际实验。例如，很多物理学的学生利用纸带、滑块和气垫导轨研究过质量、力和加速度之间的关系。物理实验允许在实践中检验理论（比如牛顿第一运动定律）。

现代科学的特征是模型、理论、实验和数值模拟。科学家们经常无法通过实际实验来验证理论，因为实验过于昂贵、过于耗时或是根本无法实现。因此，数值模拟成为了科学家们日益重要的工具，如图 1.1 所示。科学家使用数值模拟来实现模型，比较数值模拟的行为并从自然界得到的数据。科学家可以利用这些数据的差别来修订理论或是进行更多的观察。

许多重要的科学问题是非常复杂，需要功能非常强大的计算机来进行数值模拟。这些复杂的问题，经常视作为科学上的重大挑战问题。它们可以分为以下几类：

- (1) 量子化学、统计力学和相对论物理学；
- (2) 宇宙学和天体物理学；
- (3) 计算流体力学和湍流；
- (4) 材料设计和超导；

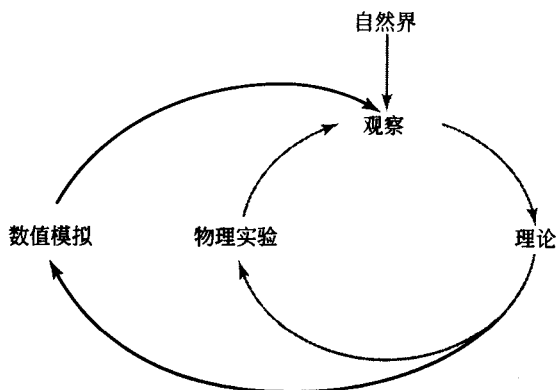


图 1.1 数值模拟方法的引入区分了经典科学方法和现代科学方法

- (5) 生物学、制药研究、基因组序列分析、基因工程、蛋白质折叠、酶活动和细胞建模；
- (6) 药物、人类骨骼和器官建模；
- (7) 全球天气和环境建模。

尽管这些重大挑战问题是在 20 世纪 80 年代才作为高性能计算进一步发展的推动力而出现的，但你也可以把整个电子计算的历史看作对高性能的追求的历史。

1.3 超级计算的进化

美国政府在开发和利用高性能计算机方面扮演了重要的角色。在第二次世界大战期间，美国军队为了加速弹道表的计算而投资研制了 ENIAC。在二战后的 30 年里，美国政府使用高性能计算机来设计核武器、破译密码，并进行其他与国家安全相关的应用。

超级计算机是所能建造的、功能最强大的计算机（随着计算机速度的增加，成为“超级计算机”的条件也会随之提高）。术语“超级计算机”是 1976 年随着 Cray-1 计算机的出现而得到广泛使用的。Cray-1 是流水线向量处理器，而不是多计算机系统，但是其计算能力可以超过每秒 1 亿次浮点运算。

超级计算机通常需要 1000 万美元以上。高成本一度意味着超级计算机几乎只能在政府所属的研究机构中找到，比如美国政府的 Los Alamos 国家实验室。

但是，随着时间的推移，超级计算机逐渐开始在政府部门以外出现了。20 世纪 70 年代末，超级计算机在资本密集型工业中开始得到应用，石油公司使用超级计算机寻找石油，汽车生产商开始使用超级计算机来改进其产品的燃油效率和安全性。

10 年后，全球数以百计的公司使用超级计算机来支持其业务，原因很简单：对很多业务来说，更快的计算能力意味着竞争优势。更快的碰撞模拟实验可以减少汽车制造商设计一辆新车所用的时间；更快的药物设计可以增加医药公司所持有的专利数。高速计算机甚至用于设计类似于一次性尿布这样的日常用品！

在过去 50 年中，计算速度有了飞速的增长。ENIAC 每秒可以执行约 350 次乘法，今天的超级计算机则要快上 10 亿倍，能够每秒执行数万亿次浮点运算。

单处理器的速度比 50 年前增长了大约 100 万倍。主要的性能增长归功于处理器时钟