国家出版基金项目
NATIONAL PUBLICATION FOUNDATION

信息与计算科学丛书—68

并行计算与实现技术

迟学斌 王彦棡
王 珏 刘 芳 编著

科学出版社

“十二五”国家重点图书出版规划项目

信息与计算科学丛书 68

并行计算与实现技术

迟学斌 王彦棡 王 珏 刘 芳 编著

科 学 出 版 社

北 京

内 容 简 介

本书系统地介绍了并行计算的基础知识和相关算法，并分别介绍了目前主流的并行编程语言 MPI、OpenMP 以及 CUDA 的相关语法、编程以及优化技巧等知识，是并行计算程序开发人员快速入门的一本较全面的教材和参考书.

本书共 6 章. 第 1 章介绍并行计算的基础知识，阐明了并行计算的起源、发展和现状以及相关的基本概念；第 2 章介绍部分基础的并行算法，包括区域分解、功能分解、流水线等六种方法，并帮助读者掌握并行算法设计的基本原则；第 3 章针对矩阵乘法、线性方程组求解、经典迭代算法的并行化、特征值求解这四类典型的数学问题，深入介绍了对应的经典的并行计算算法；第 4 章和第 5 章分别介绍了目前使用最广泛的消息传递编程语言 MPI 和共享存储并行编程语言 OpenMP 的相关知识和编程技巧；最后一章介绍了 GPU 并行加速实现技术，并重点介绍了 GPU 上使用最广泛的 CUDA 语言的相关语法、硬件架构、优化技巧以及与 MPI/OpenMP 的混合编程方法.

本书可以作为并行计算的入门指导和编程参考书，主要面向从事高性能计算的研究人员与工程技术人员，开设并行计算相关课程的高等院校与科研机构也可以选用本书作为教材.

图书在版编目(CIP)数据

并行计算与编程技术/迟学斌等编著. —北京：科学出版社，2015.6
(信息与计算科学丛书；68)
“十二五”国家重点图书出版规划项目
ISBN 978-7-03-044550-6
Ⅰ. ①并… Ⅱ. ①迟… Ⅲ. ①并行算法 Ⅳ. ①TP301.6
中国版本图书馆 CIP 数据核字 (2015) 第 122008 号

责任编辑：王丽平 / 责任校对：钟 洋
责任印制：肖 兴 / 封面设计：陈 敬

科学出版社出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com

北京佳信达欣艺术印刷有限公司印刷
科学出版社发行 各地新华书店经销
*
2015 年 6 月第 一 版 开本：720 × 1000 1/16
2015 年 6 月第一次印刷 印张：12 3/4
字数：260 000
定价：78.00 元
(如有印装质量问题，我社负责调换)

《信息与计算科学丛书》编委会

《信息与计算科学丛书》序

20 世纪 70 年代末, 由已故著名数学家冯康先生任主编、科学出版社出版了一套《计算方法丛书》, 至今已逾 30 册. 这套丛书以介绍计算数学的前沿方向和科研成果为主旨, 学术水平高、社会影响大, 对计算数学的发展、学术交流及人才培养起到了重要的作用.

1998 年教育部进行学科调整, 将计算数学及其应用软件、信息科学、运筹控制等专业合并, 定名为“信息与计算科学专业”. 为适应新形势下学科发展的需要, 科学出版社将《计算方法丛书》更名为《信息与计算科学丛书》, 组建了新的编委会, 并于 2004 年 9 月在北京召开了第一次会议, 讨论并确定了丛书的宗旨、定位及方向等问题.

新的《信息与计算科学丛书》的宗旨是面向高等学校信息与计算科学专业的高年级学生、研究生以及从事这一行业的科技工作者, 针对当前的学科前沿、介绍国内外优秀的科研成果. 强调科学性、系统性及学科交叉性, 体现新的研究方向. 内容力求深入浅出, 简明扼要.

原《计算方法丛书》的编委和编辑人员以及多位数学家曾为丛书的出版做了大量工作, 在学术界赢得了很好的声誉, 在此表示衷心的感谢. 我们诚挚地希望大家一如既往地关心和支持新丛书的出版, 以期为信息与计算科学在新世纪的发展起到积极的推动作用.

石钟慈

2005 年 7 月

前　言

近年来, 随着计算机硬件的快速发展, 采用计算机进行数值模拟计算在科学研究和实际应用中发挥着越来越大的作用, 逐渐成为科研人员与工程技术人员必不可少的研究手段和工具之一. 随着数据规模的快速增大和求解问题复杂度的提高, 仅靠单台计算机通常难以满足大规模快速计算的要求, 尤其是对于天气预报、信息检索等实时性要求较高的应用. 并行计算这一学科是为了提高计算机求解问题的速度而发展出来的. 所谓并行计算, 是指在并行机上, 将一个应用分解成多个子任务, 分配给不同的处理器, 各个处理器直接相互协同, 并行地执行子任务, 从而达到提高求解速度, 或者求解大规模应用问题的目的. 并行任务的执行需要满足并行机、并行算法的设计以及并行程序的编制这三个基本条件.

并行机作为开展并行计算的基础, 从 20 世纪六七十年代开始迅速发展. 随着计算机单个芯片的计算能力达到物理极限后, 计算机芯片的发展趋势逐渐走向多核, 出现了多核 CPU 以及多核 GPU, 并逐渐成为主流处理器芯片. 目前常用的并行机既可以是包含有多个处理器的超级计算机, 也可以是以某种方式互联的若干台独立计算机构成的同构或异构集群. 并行机的发展速度具有类似摩尔定律的特性, 其并行性将随着时间推移不断快速扩展, 这就为并行软件系统的设计和编程开发工具提出了新的挑战. 针对并行机以及并行任务的不同特性, 科研人员逐渐开发出 MPI、OpenMP 以及 CUDA 等多种并行化编程开发工具, 为并行编程提供了极大的便利. 其中 MPI 主要适用于分布式存储的可扩展的并行计算机和工作站机群; OpenMP 主要适用于共享内存多处理器系统和多核处理器等体系结构, 且与 MPI 类似, 主要适用于粗粒度任务并行计算; 而 CUDA 主要适用于包含 GPU 设备的单个计算节点, 或者与 MPI/OpenMP 协同编程以应用于包含多个 GPU 设备的集群, 主要适用于细粒度数据并行计算. 目前这些并行技术和工具已经广泛应用于石油勘探、天文计算、流体力学模拟、生物计算等领域, 大大提高了数据的处理和计算速度, 并取得了巨大的经济效益和社会效益. 随着并行机的计算能力的快速提高, 并行算法的设计和相应并行程序的编制, 以及在不同规模和类型的并行机上的扩展性和可移植性, 已逐渐成为大规模并行计算机发展和应用的主要制约因素.

伴随着高性能并行计算机在我国各大型研究机构、大学院校以及相关企事业单位逐渐普及, 并行计算已同样成为国内许多科研和工程技术人员亟需掌握的一项研究开发手段和工具. 但是, 目前国内对并行计算的基本原理、并行算法设计与实现、并行程序的性能优化, 以及各种并行编程工具的系统介绍的普及教育和推广应

用力度还略显不足, 因此一定程度上制约了并行计算技术在科学研究和工程设计中应该发挥的作用. 作者希望通过本书, 能够在科学研究和工程应用领域进一步普及并行计算技术, 以推进高性能技术在各个学科领域的应用.

作为并行计算的一本入门书籍, 本书主要面向从事高性能计算的研究人员与工程技术人员, 对并行计算的发展历程、并行计算设计的基本思想和部分经典算法进行了简单扼要的介绍, 使得读者不需要太多的计算机学科背景知识, 就能够在学习和使用的过程中逐步应用并行计算的思想和技术, 以解决工作领域遇到的各类计算问题. 书中着重对 MPI、OpenMP、CUDA 这三大主流并行化实现技术和工具进行了详细的介绍和对比. 同时, 书内附有各类技术的多个并行实现实例, 可以使读者对并行化计算方法和实现技术有全面的了解, 并快速掌握各种并行化技术的核心思想和主要编程技巧. 本书对人才培养和从事计算科学研究的人员起到一定的辅助作用, 同时为并行化算法的研究提供了新的思路, 是并行化技术入门进阶的较好的一本参考书籍.

作　者

2014 年 10 月

目　录

第 1 章　并行计算基础

近期随着计算机的快速发展, 并行计算技术已经得到广泛发展, 日常办公用计算机已经是由多核处理器组成的并行计算机, 因此, 并行计算机对于今天的每个人来说, 已经并不陌生. 如何有效地让这些计算机发挥其性能, 使得物尽其用, 就需要了解这些计算机的特点, 设计适合并行计算机执行的算法和相应的程序. 本章将重点介绍为什么需要并行计算、并行计算机的发展历程、并行算法复杂性分析以及并行计算的基本概念. 目的是让读者对并行计算有一个基本认识, 掌握一定的理论基础.

1.1　什么是并行计算

并行计算 (parallel computing) 是相对于串行计算提出来的, 其基本思想是通过多个处理器共同解决同一个计算问题, 即每一个处理器单独承担整个计算任务中的一部分内容. 因此计算任务的分解就是并行计算中首要考虑的关键问题, 目前常见的分解方法可大体分为时间和空间上的并行, 一般情况下计算问题在时间上具有相互关联的特点, 所以关于时间并行方法往往采用流水线 (详见 2.4 节流水线技术) 的方式实现, 近期也有部分研究人员提出通过一种拟合方法来完成时间上的并行[1], 不过此类方法往往对算法上有一定要求. 空间上并行是目前绝大多数计算问题的并行解决方案, 即在同一时刻将整个计算任务按照某种规则在空间上进行分解, 在下一时刻开始前同步相关的参数, 最典型的代表就是区域分解算法 (详见 2.2 节区域分解方法). 此外, 分解计算任务还要考虑负载均衡、通信量等问题, 它们直接关系到并行程序的效果.

通过上面的介绍, 我们对并行计算有了一个简单了解, 下面给出并行计算的常用定义.

定义 1.1　*并行计算是指在并行机上, 将一个计算问题分解成多个子任务, 分配给不同的处理器, 各个处理器之间相互协同, 并行地执行子任务, 从而达到加速求解, 或者求解大规模应用问题的目的.*

开展并行计算, 必须具备三个基本条件[2]:

(1) 并行机. 并行机包含多个处理器核心, 这些处理器核心通过特定硬件相互连接, 相互通信.

(2) 应用问题必须具有并行度. 也就是说, 应用可以分解为多个子任务, 这些子任务可以并行地执行. 将一个应用分解为多个子任务的过程, 称为并行算法的设计.

(3) 并行编程. 在并行机提供的并行编程环境上, 具体实现并行算法, 编制并行程序, 并运行该程序, 从而达到并行求解应用问题的目的.

下面通过一个数组求和的例子来给出并行算法设计的思路.

例 1.1　*并行计算求和问题*

$$S = \sum_{i=0}^{n-1} a_i \tag{1.1}$$

假设 $n = 4 \times m$, 记 $S_0 = \sum_{i=0}^{m-1} a_i$, $S_1 = \sum_{i=m}^{2m-1} a_i$, $S_2 = \sum_{i=2m}^{3m-1} a_i$, $S_3 = \sum_{i=3m}^{n-1} a_i$, 则有 $S = S_0 + S_1 + S_2 + S_3$.

因此, 计算 S 可以并行执行, 亦即 $S_i(i = 0, 1, 2, 3)$ 可以同时计算出, 进一步如果令 $S_{00} = S_0 + S_1$ 和 $S_{01} = S_2 + S_3$, 可以同时计算 S_{00} 和 S_{01}, 最后再计算 $S = S_{00} + S_{01}$.

1.2　为什么需要并行计算

翻开计算机发展历史, 计算性能的提升主导着整个计算机技术发展, 提高传统计算机的计算速度一方面受到物理上光速极限和量子效应的限制, 另一方面计算机器件产品和材料的生产受到加工工艺的限制, 其尺寸不可能做得无限小, 因此目前主流处理器已经向双核、四核、多核以及众核的方向发展. 此外, 随着计算在科学研究和实际应用中发挥越来越大的作用, 人们对计算已经产生了依赖, 将数值模拟作为许多决策的依据. 现在人们已经习惯将计算作为科学研究的第三种手段, 和传统的科学研究的理论方法和实验方法并列.

并行计算是人们为了提高求解问题的计算速度而提出的, 从 20 世纪 60 年代开始, 逐渐得到丰富和发展. 众所周知, 今天的单个计算机芯片的处理能力已经可以达到每秒万亿次的计算能力, 这在 21 世纪初, 其计算性能相当于一台巨型计算机. 纵然单个计算机的计算能力已经非常强大, 仍然无法满足有很多科学与工程计算问题的计算需求, 因此, 并行计算机是解决这些问题不可或缺的重要工具. 也正是因为科技进步的要求, 并行计算正逐步渗透到科学研究与工程应用之中.

当今随着大数据时代的来临, 对数据进行处理也同样离不开并行计算, 可以说, 并行计算是无处不在的. 比如, 我们要了解我们生存的地球, 就需要知道天气变化、环境变化、地震运动等自然规律. 在充分的数据分析基础上, 科学家们提出了一系列数学物理模型, 使得我们今天能够对天气进行预报, 对地震进行预测, 对环境变

化进行研究, 结果的获得就需要并行计算的支持. 尽管目前的预报和预测还不能让我们满意, 但已经取得了重要进展. 现代天气预报技术已经趋向于成熟, 预报的准确性已经很高, 进入了数值天气预报阶段, 这项工作也正是有现在的高性能计算机的支持, 才能够让我们掌握更多大自然的发展规律, 为人类的明天提供更加可靠的数据. 还有很多科学研究领域, 诸如, 天体物理中关于宇宙的形成、暗物质作用、材料科学中晶体结构、流体力学中湍流、基因组学中序列拼接、蛋白组学中结构分析等研究领域与问题, 同样需要高性能计算机的强大计算能力.

在工业设计应用中, 也同样离不开并行计算技术的支持. 比如我们如今每天都需要的出行工具汽车, 无论是发动机的设计, 还是外观设计均需要进行大量的数值模拟. 特别是汽车碰撞试验, 不能每次都使用实际汽车来进行试验, 一是成本高, 重要的是还有安全因素. 再比如与我们生命息息相关的药物研制, 需要用大量的实验数据来支撑, 为了获取这些重要的数据, 完全由实验获取是非常困难的. 一方面每次实验能够得到的数据有限, 也存在着危险性; 另一方面, 为获取大量数据, 需要进行多次实验, 时间与经费也难以承受. 因此, 在药物研制领域, 已经越来越多的使用超级计算机进行药物筛选, 极大地缩短了药物研发周期和新药研制成本. 还有很多工业领域的产品设计都依赖高性能计算机, 正如我们熟知的飞机设计、高速列车设计、建筑物设计等都需要进行数值模拟, 为最终决策提供重要的依据.

在国家安全方面, 高性能计算机越来越发挥着不可替代的作用. 众所周知, 密码已经是我们生活中习以为常的防护手段, 登录计算机、建立银行账号等都需要与密码配合, 在一定程度上使我们的信息和财产等得到了保护. 由此可见, 密码是非常重要的, 如果泄露, 后果不堪设想. 在信息传递过程中, 为了保证信息安全, 通常采用加密手段, 使信息不易被获取. 然而这种加密手段也可以通过强大的计算能力来解决, 主要取决于计算能力是否能够在短时间内获得一个大整数的 2 个素数因子. 再有, 许多案件的破获是源于指纹信息, 通过对指纹进行比对, 可以非常准确地确定犯罪分子. 指纹比对的过程, 不可能是指纹专家仅凭肉眼来识别, 只能借助于高性能计算机在浩瀚的指纹数据库中搜索, 锁定目标之后, 再由指纹专家进行确认.

20 世纪 90 年代, 美国提出了 HPCC 计划, 是为了解决一大批巨大挑战性问题, 其中包括: 天气与气候预报, 分子、原子与核结构, 大气污染, 燃料与燃烧, 生物学中的大分子结构, 新型材料特性, 国家安全等有关问题. 而近期内要解决的问题包含: 磁记录技术, 新药研制, 高速城市交通, 催化剂设计, 燃料燃烧原理, 海洋模型模拟, 臭氧层空洞, 数字解剖, 空气污染, 蛋白质结构设计, 金星图像分析和密码破译技术. 为此, 世界许多国家都大力开展高性能计算机系统的研制, 目的是为在技术上获得制高点. 我国一直重视高性能计算机系统的研制, 天河一号在 2010 年世界 TOP 500 排行榜上位列第一名, 2013 年, 我国天河二号再次问鼎世界第一.

1.3　并行计算机的发展

并行计算机从 20 世纪 70 年代开始, 到 80 年代蓬勃发展和百家争鸣, 90 年代体系结构框架趋于统一, 随着近年来机群技术的快速发展, 并行机技术日趋成熟. 本节以时间为线索, 简介并行计算机发展的各个阶段, 以及各类并行机的典型代表和它们的主要特征.

1972 年, 世界上诞生了第一台并行计算机 ILLIAC IV[3], 它含 32 个处理单元, 每台处理机拥有局部内存, 为 SIMD 类型机器, 对大型流体力学程序, ILLIAC IV 获得了 2 到 6 倍于当时性能最高的 CDC 7600 机器的速度, 并在 1975 年成为可供研究人员通过网络使用的大型计算机. 70 年代中后期, 出现了 Cray-1 为代表的向量计算机, 2002 年日本研制的世界上先进的高性能计算机系统 —— 地球模拟器的处理器就是采用向量机.

进入 80 年代, MPP 并行计算机开始大量涌现, 这种类型的计算机早期的典型代表有 iPSC/860, nCUBE-2, Meiko. 我国也在这个时期, 研制成功了向量计算机银河–1.

从 90 年代开始, 主流的计算机仍然是 MPP, 例如: IBM SP2, Cray T3D, Cray T3E, Intel Paragon XP/S, 中国科学院计算技术研究所“曙光 1000”等.

目前主要采用的计算机系统结构为:

(1) 机群系统[4].

(2) DSM, SMP 系统[5].

(3) MPP 系统[6].

(4) 星群系统[7].

这里的星群实际上也是一种机群, 它包含了异构机群, 每个计算结点可以是不同结构的 SMP、MPP 等构成. 在 2013 年的高性能计算机系统 (表 1.1) 中, 以混合结构完成的系统取得了飞速发展. 在这一年 TOP500 前 3 中的 2 台计算机系统都是混合结构的. 混合结构能够取得很高的峰值速度, 若在实际应用中能够发挥作用, 需要进行大量艰苦的努力.

表 1.1　2013 年 TOP500 计算机系统的前 10 名 (GFLOPS)

计算机系统	国家	年份	核数	性能
Tianhe-2, TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P	中国	2013	3120000	33862700
Titan, Cray XK7, Cray Gemini Opteron 6274 16C 2.200GHz, interconnect, NVIDIA K20x	美国	2012	560640	17590000

续表

计算机系统	国家	年份	核数	性能
Sequoia, BlueGene/Q, Custom Power BQC 16C 1.60 GHz	美国	2011	1572864	17173224
K computer, Tofu interconnect SPARC64 VIIIfx 2.0GHz	日本	2011	705024	10510000
Mira, BlueGene/Q, Custom Power BQC 16C 1.60GHz	美国	2012	786432	8586612
Stampede, PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P	美国	2012	462462	5168110
JUQUEEN, BlueGene/Q, Custom Power BQC 16C 1.600GHz	德国	2012	458752	5008857
Vulcan, BlueGene/Q, Custom Power BQC 16C 1.600GHz	美国	2012	393216	4293306
SuperMUC, iDataPlex DX360M4, Infiniband Xeon E5-2680 8C 2.70GHz	德国	2012	147456	2897000
Tianhe-1A, NUDT YH MPP, NVIDIA 2050 Xeon X5670 6C 2.93 GHz	中国	2010	186368	2566000

我国高性能计算机的研制水平一直处于世界领先行列, 这对我国开展科学研究、促进技术进步, 所起到的巨大推动作用是毋庸置疑的. 然而我们必须清醒地认识到, 在使用高性能计算机方面, 与先进国家相比, 还有巨大差距. 希望我们的工作能够为仁人志士投入到并行计算行列提供力所能及的帮助.

1.4 并行算法复杂性分析

在数值计算中, 我们通常需要预先知道一个算法在计算机上能够何时完成, 从理论上就可以得出哪些算法具有优势. 因此, 需要建立分析算法复杂性的手段, 从而可以对算法进行定量评估. 这里需要首先明确什么是算法复杂性, 再者什么是并行算法复杂性.

定义 1.2 *算法复杂性是由四则运算次数和存储空间大小两部分组成.*

在接下来的论述中, 只针对四则运算次数进行分析, 这也是算法复杂性的重点. 下面用两个例子说明如何得到一个算法的复杂性.

例 1.2 *计算 x^n 的复杂性.*

首先, 我们需要给出计算 x^n 的算法, 针对算法给出计算复杂性. 对于一个问题来说, 通常其计算复杂性是确定的, 因此我们在算法设计时就应该寻找最佳的计算复杂性算法, 从而达到快速计算之目的. 那么, 怎样计算 x^n? 比如 $n = 13$, 可以计

算 $x^2 = x \times x$, $x^4 = x^2 \times x^2$, $x^8 = x^4 \times x^4$, $x^5 = x^4 \times x$, 最后计算 $x^{13} = x^8 \times x^5$, 一共需要 5 次计算. 我们的算法就是基于这种方式, 按照 2 的幂次进行计算. 对于给定的一个整数 n, 其 2 进制表示可以记为: $n = 2^k + a_{k-1}2^{k-1} + \cdots + a_1 2 + a_0$, 其中 $a_i = 0$ 或者 $a_i = 1$, $k = \log_2 n$. 据此可以给出计算方法如下:

算法 1.1　快速计算 x^n 的算法.

$y = 1$;
for $i = 0$ to k-1 do
　if a[i] = 1 do
　 $y = y * x$;
　end{if}
　$x = x * x$;
end{for}
$y = y * x$;

因此可以得出, 当所有的 a_i 均非零时, 这个算法的计算复杂性为 $2k + 1$, 即 $2\log_2 n + 1$, 通常亦称之为计算复杂性为 $O(\log n)$. 通过此算法可以看到, 即使一个简单的问题, 其计算方法也可以很复杂, 但该问题并不适合并行计算.

例 1.3　计算矩阵乘向量的复杂性.

这是一个科学计算中适合并行计算的常见问题, 即矩阵乘向量 $y = Ax$, 其中矩阵 A 是 $m \times n$ 阶的, y 与 x 分别是 m 和 n 维向量. 记矩阵 A 的第 i 列为 A_i, 则我们有如下计算方法:

计算 $y = Ax$ 的算法.

$y = 0$;
for $i = 0$ to n-1 do
　$y = y + A[i] * x[i]$;
end{for}

这里的基本运算是一个列向量 y 加一个常数 $x[i]$ 乘另一个列向量 $A[i]$, 通常称此运算为“链三元”. 这个算法非常简单, 计算复杂性也可以通过如下的方式计算出来. 一个 m 维的链三元运算计算复杂性是 $2m$, 而我们的算法一共需要 n 个链三元运算, 因此, 计算 $y = Ax$ 算法复杂性为 $2mn$. 如果矩阵为方阵, 则计算复杂性为 $2n^2$, 亦即需要 $O(n^2)$ 的计算量, 也就是我们所熟知的计算矩阵乘向量的复杂性.

以上我们用 2 个例子说明如何计算算法的复杂性, 接下来我们来讨论什么是并行计算方法的计算复杂性. 为简单起见, 这里我们也只考虑计算复杂性, 不考虑通信与存储复杂性.

定义 1.3　并行计算复杂性是所有进程中的最大计算次数.

如例 1.1, 假设用 $p = 2^k$ 个处理器计算 $n = 2^k$ 的求和问题, 这时每个处理器