国外经典教材·计算机科学与技术

PEARSON
Prentice Hall

Fundamentals of Parallel Processing

并行处理基本原理

(美) HARRY F. JORDAN
GITA ALAGHBAND 著

迟利华 刘杰 译

清华大学出版社

国外经典教材•计算机科学与技术

并行处理基本原理

(美) Harry F. Jordan
Gita Alaghband 著

迟利华 刘 杰 译

清华大学出版社

北 京

EISBN: 0-13-901158-7

图书在版编目(CIP)数据

并行处理基本原理/(美)乔丹(Jordan,H.F.)，(美)阿拉格邦德(Alaghband,G.)著；迟利华，刘杰译.
—北京：清华大学出版社，2004.10
书名原文：Fundamentals of Parallel Processing
(国外经典教材•计算机科学与技术)
ISBN 7-302-09003-3

Ⅰ.并… Ⅱ.①乔…②阿…③迟…④刘… Ⅲ.并行处理－教材 Ⅳ.TP274

中国版本图书馆 CIP 数据核字(2004)第 067215 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
组稿编辑：曹 康
文稿编辑：杜一民
封面设计：久久度文化
版式设计：康 博
印 装 者：北京鑫海金澳胶印有限公司
发 行 者：新华书店总店北京发行所
开 本：185×260 **印张**：29 **字数**：742 千字
版 次：2004 年 10 月第 1 版 2004 年 10 月第1 次印刷
书 号：ISBN 7-302-09003-3/TP • 6363
印 数：1～4000
定 价：52.00 元

出 版 说 明

近年来，我国的高等教育特别是计算机学科教育，进行了一系列大的调整和改革，急需一批门类齐全、具有国际先进水平的计算机经典教材，以适应当前我国计算机科学的教学需要。通过使用国外先进的经典教材，可以了解并吸收国际先进的教学思想和教学方法，使我国的计算机科学教育能够跟上国际计算机教育发展的步伐，从而培育出更多具有国际水准的计算机专业人才，增强我国计算机产业的核心竞争力。为此，我们从国外知名的出版集团 Pearson 引进这套“国外经典教材·计算机科学与技术”教材。

作为全球最大的图书出版机构，Pearson 在高等教育领域有着不凡的表现，其下属的 Prentice Hall 和 Addison Wesley 出版社是全球计算机高等教育的龙头出版机构。清华大学出版社与 Pearson 出版集团长期保持着紧密友好的合作关系，这次引进的“国外经典教材·计算机科学与技术”教材大部分出自 Prentice Hall 和 Addison Wesley 两家出版社。为了组织该套教材的出版，我们在国内聘请了一批知名的专家和教授，成立了一个专门的教材编审委员会。

教材编审委员会的运作从教材的选题阶段即开始启动，各位委员根据国内外高等院校计算机科学及相关专业的现有课程体系，并结合各个专业的培养方向，从 Pearson 出版的计算机系列教材中精心挑选针对性强的题材，以保证该套教材的优秀性和领先性，避免出现“低质重复引进”或“高质消化不良”的现象。

为了保证出版质量，我们为该套教材配备了一批经验丰富的编辑、排版、校对人员，制定了更加严格的出版流程。本套教材的译者，全部来自于对应专业的高校教师或拥有相关经验的 IT 专家。每本教材的责编在翻译伊始，就定期不间断地与该书的译者进行交流与反馈。为了尽可能地保留与发扬教材原著的精华，在经过翻译、排版和传统的三审三校之后，我们还请编审委员或相关的专家教授对文稿进行审读，以最大程度地弥补和修正在前面一系列加工过程中对教材造成的误差和瑕疵。

由于时间紧迫和受全体制作人员自身能力所限，该套教材在出版过程中很可能还存在一些遗憾，欢迎广大师生来电来信批评指正。同时，也欢迎读者朋友积极向我们推荐各类优秀的国外计算机教材，共同为我国高等院校计算机教育事业贡献力量。

清华大学出版社

前　言

致学生

老师在讲授计算方法时，通常采用串行的观点。算法按照计算步骤进行组织，程序按照逐条指令的方式进行编写，而计算机则按照一系列的微操作指令执行。尽管串行模式也能够解决问题，但是以并行的方式让计算机同时执行多个操作可以极大地获得性能上的提高。提高计算速度的基本方法有两种：一是提高硬件潜在的时钟频率；二是让多个操作并行执行。其中利用并行操作来提高计算速度是更有前景，因为当任务变得庞大时，将会有更多的操作可以并行执行。要实现这种潜在的并行性，必须让以下三方面同时起作用：一是计算方法必须涉及许多相互独立的操作；二是编程语言必须可以定义并行操作，或者自动地找出可以并行执行的操作；三是运行并行程序的计算机的体系结构必须能够同时执行多个操作。

并行处理是算法设计、程序设计语言和计算机体系结构三者相结合的产物，其目的是为了计算机系统能够更快地完成对应用问题的计算。并行处理基本原理的出现正是源于对上述计算主题的理解以及它们之间的相互结合所带来的高性能。为了较好地理解并行处理，必须具备一些计算机设计和体系结构、编程语言和机器代码的基础知识，而且还必须具备算法结构的基础知识。本书的一些章节集中讨论了体系结构、编程语言和算法，但它们之间并没有明显的区分。这 3 部分共同构成并行处理这门学科的概念基础。我们希望读者具备算法和编程的基本知识，为了说明并行处理的真正目的(为了获得更高的性能)，必须知道计算机是在机器语言级执行程序的，同时要求读者理解构成计算机体系结构的各个硬件部件的组织方式。此外，在该领域中的一些基本经验也是阅读本书的必备前提。

致教师

本书的目的就是要全面而综合地介绍并行处理的基本原理。要想学习知识既有深度又有广度，同时掌握成功设计和开发并行应用程序所需的专门技术，计算机体系结构、算法和语言知识的整合是关键。本书的组织与编写在这些主题领域之间建立起紧密的联系。在讨论了算法设计之后，本书还进一步讨论了每一种算法设计对于并行体系结构性能的影响。

技术的不断革新，新的体系结构、新式程序设计语言和新系统的不断涌现要求我们对并行处理有一个基本的了解。本书注重对于基本概念的阐述，而不是对于大量最新发展趋势的说明。本书的结构安排十分合理，每一个章节都是前一章节的自然延续。本书在论述每一主题时，都会尽早提出研究主题，从而促使读者进一步地学习，并对研究主题有一个全局的把握，明确学习什么和怎样学习。如果不是把最新的体系结构、编程语言和系统作为讲授基本概念的一个工

具，而仅仅是为了学习这些时髦的东西，那么学习过程将很困难，而且您会发现它们很快就会过时。如果读者没有达到忘我的境界，并且对研究主题仍感困惑的话，那么要想学到精髓是很困难的。在任何情况下，在深入到核心内容之前，剥开一些表层的附加信息和特性是很有必要的。例如，一种编程语言有必要提供众多的结构体吗？是不是只有一些是必需的，而另外一些仅仅是为了方便使用而额外提供的呢？实现它们是为了满足特定体系结构的性能要求，还是为了提供可移植性？这些结构体的实现具有相关性吗？它们的性能会随着计算机体系结构的不同而相差很大呢？仅仅通过了解机器和语言，要想完全理解这些权衡和概念的内涵是不可能的。一旦理解了这些基本概念，就可以把它们应用到任何体系结构、系统和语言中。

并行处理是一个相对年轻的学科，我们相信，该学科已经能够从一些独立的系统中分离出来，而成为一个相对独立和值得探讨的基础学科。我们主要通过体系结构的特点、系统的特性、语言的结构体、算法的设计和实现来展示一些基本原理，而对于这些，我们尽可能地以一种与特定体系结构、系统和编程语言无关的方式进行讲解。在有些情况下，我们涉及到一些机器、语言或者系统的原型，因为这样便于我们引入基本的概念。但是，作为主要的实例，我们有意避免把每一个主题都扩展到一些具体的机器或语言上，这样做使我们可以把主要的精力放在基本原理上。

尽管本书不是并行程序设计的教科书，但是对于书中引入的每一类主要的并行概念都是通过一种具体的编程语言来表述的，书中选择 Fortran 作为基本语言。很多研究并行处理的文献都是基于 Fortran 的，大量并行处理科学的计算程序和程序员都使用 Fortran。另外，Fortran 是一种接近机器级而又很简单的高级语言，相对于具有复杂特性的、用户界面友好的高级语言而言，在各种不同体系结构的计算机上，Fortran 更便于观察和解释语句的执行效果，Fortran 程序员对编程模式和程序的设计、实现以及执行能有更好的控制。Fortran 是静态语言。因此，相对于动态语言或者提供动态属性的语言而言，Fortran 程序员不必太多关注它们的高级特性和并行性能的影响。语言的简单性更有利于我们重点关注并行概念和并行结构体，Fortran 支持多维数组的计算，这对于向量处理具有特别重要的意义。在整本书中，我们都使用同一种基本语言，这有利于表达的一致性，读者不必在语言之间来回切换，这更有利于集中精力关注并行问题。

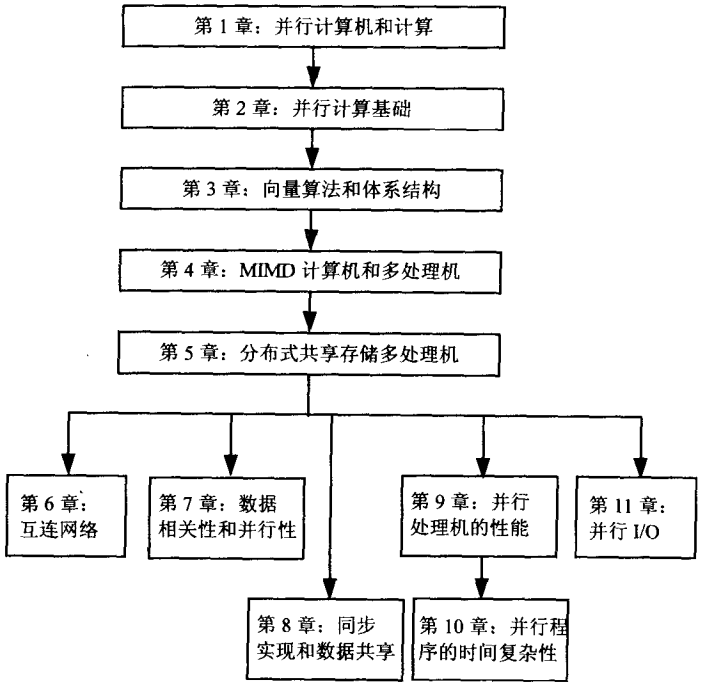
本书根据并行处理领域多年教学和科研经验组织和设计的，主要读者对象是计算机科学或者计算机工程与应用专业高年级学生和研究生。通过本书的学习，读者将能够胜任并行应用程序的设计和实现，评估并行程序和体系结构的性能，更重要的是，通过独立学习新的并行环境，能够培养学生独立开发的技能。教师主要的任务之一就是培养学生在感兴趣的领域继续发展和取得进步的能力。本书还将给教师提供一些综合性的材料，这些综合性的材料能够培养学生更具创造力并获得更大的成功。

本书主要内容

前 7 章的内容可以作为并行处理第一学期的课程，这 7 章比较深入地介绍了并行算法设计、向量、多处理机、数据流结构、针对各种机器的并行程序设计语言、同步和通信机制、互连网络、数据相关性和编译器的优化技术。其余的 4 章主要是对本书第一部分的内容进行了更深入的探讨。主要讨论了各种同步和通信方式的实现、其实现方法对性能的影响、机器体系结构和

程序性能的解释、程序的行为对性能的影响以及并行 I/O。这一部分可以作为第二学期的课程，学生可以学习到如何分析机器体系结构、并行程序和并行系统、并且能够深入地理解各部分对整体性能有何影响。从第 8 章至第 11 章所讨论的主题中，我们可以得到许多高级研究项目的思想。

下面的图显示了各章之间的相互关系，以及它们所涵盖的专业领域的重点。



各章主要内容

第 1 章简要回顾了并行性在计算机体系结构中的发展历程。该章介绍了向量处理、多处理机和算法中的并行操作的基本概念，为后续章节的内容建立了一个基本的框架。

第 2 章介绍了数据相关性的一些重要思想。使用前缀计算(prefix computation)来说明算法的一些特性，这些特性使得利用不同的计算方法来完成同一个计算问题时，所获得的并行性是不同的。

第 3 章介绍了多个数据项并行执行同一操作的应用。该章引入了对线性代数中一些简单算法的讨论，同时给出了一个机器语言级的体系结构，其中包含了向量操作。之后还讨论了 Fortran 90，Fortran 90 是一种对机器级向量处理特性具有高级支持的语言。最后还对流水向量(pipelined vector)处理也进行了讨论。

第 4 章简要讲述了多处理机体系结构的组织，同时说明了共享存储器和分布式存储器多处理机的区别。并通过描述对串行程序设计的扩展，进一步讨论了共享的存储器，因为必须利用它来协调多个进程共同完成一个任务。利用扩展的 OpenMP Fortran 来说明高级结构体(construct)，这些高级结构体是用来支持共享存储器多处理机的计算的。该章也介绍了具有流水功能的 MIMD 或者是多线程体系结构的基本知识。

第 5 章描述了分布式存储器多处理机，在这样一个体系结构中，通过消息传递的观点主要讲解了数据通信的主导作用。介绍了消息的显式发送与接收的有关编程问题，并且利用消息传递接口(MPI)来说明支持这种程序的高级语言。介绍了与分布式共享多处理机相关的 Cache 一致性的基本概念。

第 6 章比较深入地讨论了互连网络，包括用来连接向量处理机的互连网络以及连接共享存储器和分布式存储器多处理机的互连网络，比较了动态网络和静态网络的特点，同时比较了各种网络拓扑结构以及这些拓扑结构的特点，讨论了在 NYU 巨型机中得到应用的互连网络。

第 7 章中有关数据相关性的讨论是很重要的，它是从并行算法的结构过渡到程序的结构的基础。它涉及到代码的优化技术和一些与编译器编写者相关的主题，编译器编写者担负着为并行计算机生成代码的任务。该章还介绍了数据流语言和体系结构的概念，有了这些概念，就可以从编程语言和机器中消除不必要的相关性。

第 8 章扩展了在第 4 章讨论共享存储器时提到的同步概念，并把其和第 5 章中强调的数据传输的观点结合起来，深入地探讨了同步中的关键问题，涉及到了协作通信的同步、管理共享事务、等待机制、以及如何证明一个同步机制是可以正确地实现的。

第 9 章集中讨论了性能问题，这些问题在前面的章节中时常被提到。该章讨论了各种性能模型，通过对统计数据和实际系统的实例研究说明了它们的作用。还讨论了并行机制的不同调度和不同实现对运行效果的影响。

第 10 章讨论了对并行程序执行的性能与并行程序的一些局部特性的关系。通过对实际系统的实验来说明性能特性模型，从单个高速缓存系统，具有分布式高速缓存的多处理机系统到消息传递系统，都进行了局部性测试。

第 11 章对 I/O 操作并行性的多个方面进行了讨论。对作为并行 I/O 硬件的并行访问磁盘阵列(RAID)进行了讲解，介绍了 I/O 操作的相关性，讨论了对文件的并行输入输出方法，最后，利用 MPI-IO，讨论了对多处理机中多个 I/O 操作的并行性。

目　　录

第1章　并行机和并行计算

为了使计算机的执行速度更快，让计算机同时做多件事情来减少执行时间显然是一种可行的技术。令人惊讶的是，并行计算机和并行计算并没有引起更多人的注意，令人关注的是许多有关计算的科学技术都是围绕着一种顺序化的，一次做一件事情的模式来进行算法设计、程序编写和机器设计的。并行处理在一个普遍顺序的思想之中得到了发展，经过很长时间之后，它才作为一个指导算法设计、机器设计和编程语言设计的高级原理而出现。

并行处理学科实际上是一个这样的主题：就是要把算法的需求和体系结构的性能结合起来。对于并行体系结构的研究而言，如果并行算法不能够对大规模运算进行高速并行处理，那么并行体系结构的研究是没有意义的。同时，如果没有可预见的能够实现的体系结构来真正地支持这种并行算法的话，为了决定哪些操作能够潜在地并行执行而进行的算法研究也就没有意义。并行处理通过并行编程语言把并行算法和并行体系结构结合起来进行研究，所以，在接下来的内容中，我们并不打算把并行处理分为并行体系结构、并行编程语言和并行算法来分别进行研究，相反，我们将交叉地来讨论并行计算的各个方面，这样更能强调它们之间相互关系的重要性。

经常带着一些具有普遍性的问题来阅读本章是很有益的，执行并行操作的多计算部件的实质是什么？怎样才能写出良好的算法，这种算法能够规定以并行的方式来执行一些操作，而以串行的方式来执行另外一些操作？针对一个具体计算的特定算法，它的哪些特征会对操作能否并行执行产生影响？

1.1　并行体系结构的发展历程

我们已经分析了并行性的几种不同的形式，它们可以通过同时处理多件事情来更快地解决一些问题。可以根据它们与标准串行计算模式的关系来描述这一内容，在这种模式中，在一个时间片内可以同时执行加、减、乘、除或条件分支中的一个操作。并行性的一种主要划分方法是基于多个数据项的并行性与多个操作的并行性。数据变换控制流在计算机科学领域中出现的时间不长，相对而言，数值分析中对一个固定操作集的数据变换更好理解，因此，从串行程序领域到并行程序领域的过渡中，不考虑程序的控制流，而直接并行地对多个独立的数据项进行操作是可能的。Flynn(见参考文献注解[110])根据计算机中指令和数据的并行状况把计算机体系机构分为四类，这种分类是依据指令流和数据流数目的不同而进行的。体系结构是通过把单指令(SD)、多指令(MI)与单数据(SD)、多数据(MD)两两组合起来进行分类的。这样，原来的冯•诺依曼体系结构就被分为 SISD(单指令单数据流)体系结构，向量计算机是 SIMD(单指令多数据流)体系结构，因为一条指令流可以同时控制多个数据的执行；多处理机则是 MIMD(多指令多数据流)体系结构。尽管 MISD 可能适合少量的研究性的机器，但是这种组合不是太有用。

在下面的内容中，我们将从计算机体系结构和系统的角度来简要地介绍一下计算机速度提

高的发展历程。

1.1.1　串行计算机中的并行性

被划分为 SISD 的冯•诺依曼体系结构把计算分为一串顺序的机器操作，机器的执行是每次执行一个操作。如图 1-1 所示，这种早期的计算机的主要组成部件包括：控制部件(CU)、算术逻辑部件(ALU)和存储器(M)，控制部件和算术逻辑部件合在一起就是中央处理器(CPU)。在这里介绍这些，主要目的不是要谈及具体的实现技术，而是要对这些操作进行一些高级描述，因为，这样可以较好地过渡到对并行性的讨论。如图 1-2 所示，计算机从启动一直到关机，都在进行取指令和执行指令的循环。

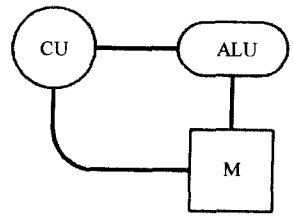


图 1-1　典型的冯•诺依曼体系结构

(1) 取指令
(2) 指令译码和程序计数器加 1
(3) 有效操作数地址计算
(4) 取操作数
(5) 执行
(6) 存储结果

图 1-2　冯•诺依曼计算机的简单取指令与执行周期

为了提高 SISD 计算机的执行速度，人们对早期的体系结构进行了很多改进，中断机制允许 CPU 操作与慢速外设重叠执行，当允许 CPU 和 I/O 处理机直接访问存储器时，输入/输出(I/O)处理机就使得快速 I/O 设备和 CPU 可以并行操作，如图 1-3 所示。在多任务的系统中，多个程序可以交替地使用单个 CPU，这样可以减少 CPU 的闲置时间，使系统的吞吐率和 CPU 的利用率达到最大化。由于几个程序以这种方式交替执行，所以操作系统必须要管理这些并发进程，这些进程都处于其执行过程的不同阶段。由于 I/O 处理机使得存储器块在主存储器和辅助存储器之间的高速传递成为可能，程序的局部性原理也要求对虚拟存储器进行管理，在这样的系统中，应用程序在所需的存储空间比系统存储器大得多的情况下依然能完成执行过程。虚拟存储器和多道程序的结合产生了多用户和分时计算环境。

流水技术是一种使得操作能够并行执行的强大技术，它把并行性的概念引入到了计算机体系结构中。通过一些额外的硬件，图 1-2 中的取指令执行周期中的步骤可以重叠执行。图 1-4 显示了一种可能的流水方式的取指令/执行周期。不同的指令按照它们进入的顺序通过不同的流水段，当多条指令顺序地执行时，其中一些指令正在不同的流水段中同时被处理。在这个例子中，共使用了四个时间单元来完成第一条指令(从进入流水线时开始)。但是，在流水线初始化以后，在每个流水段都有一条指令完成。在图 1-5 中，为了表明流水线的效率，把非流水线与