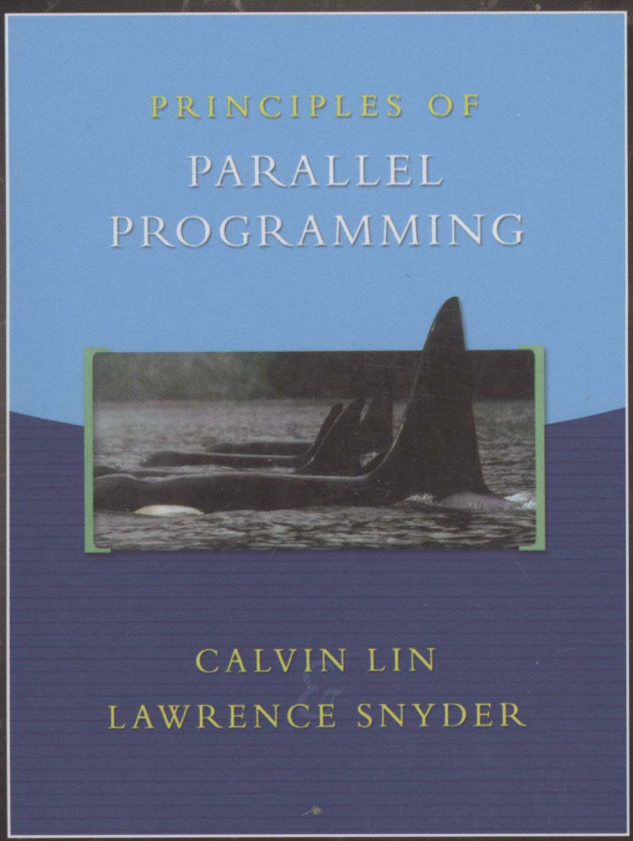
HZ BOOKS
华章教育
PEARSON
Addison Wesley

计 算 机 科 学 丛 书

并行程序设计原理

(美) Calvin Lin 得克萨斯大学奥斯汀分校 Lawrence Snyder 华盛顿大学西雅图分校 著 陆鑫达 林新华 译



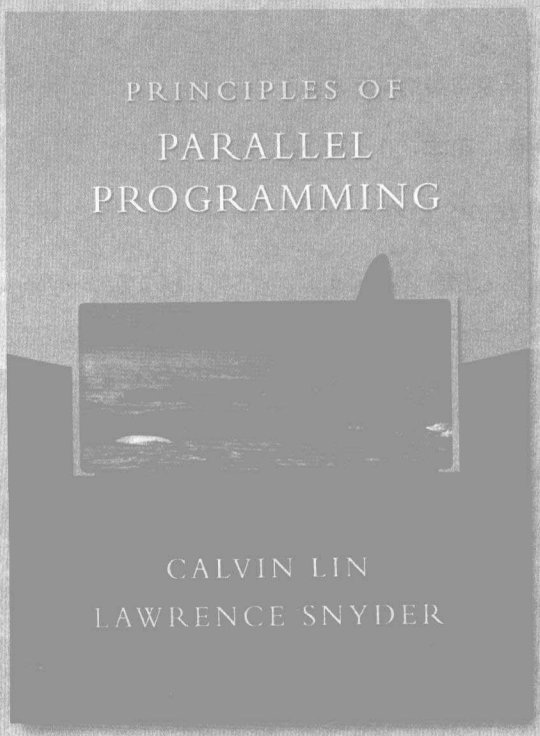
Principles of Parallel Programming

机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

并行程序设计原理

(美) Calvin Lin 得克萨斯大学奥斯汀分校 Lawrence Snyder 华盛顿大学西雅图分校 著 陆鑫达 林新华 译



Principles of Parallel Programming

机械工业出版社
China Machine Press

本书内容新颖，涉及现代并行硬件和软件技术，包括多核体系结构及其并行程序设计技术。本书侧重论述并行程序设计的原理，并论述了并行程序设计中一些深层次问题，如可扩展性、可移植性以及并行程序设计应遵循的方法学等。

本书是高等院校计算机专业高年级本科生或低年级研究生的理想教科书，同时也是专业程序员从事并行程序设计的理想入门书。

本书版权登记号：图字：01-2009-1336

图书在版编目（CIP）数据

并行程序设计原理 /（美）林（Lin, C. ），（美）斯奈德（Snyder, L. ）著；陆鑫达等译. —北京：机械工业出版社，2009.7

（计算机科学丛书）

书名原文：Principles of Parallel Programming

ISBN 978-7-111-27075-1

Ⅰ.并… Ⅱ.① 林… ② 斯… ③ 陆… Ⅲ.并行程序-程序设计 Ⅳ.TP311.11

中国版本图书馆CIP数据核字（2009）第070624号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：刘立卿

三河市明辉印装有限公司印刷

2009年7月第1版第1次印刷

184mm × 260mm · 15.75印张

标准书号：ISBN 978-7-111-27075-1

定价：45.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章分社较早意识到“出版要为教育服务”。自1998年开始，华章分社就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出Andrew S. Tanenbaum，Bjarne Stroustrup，Brain W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章分社欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

华章教育

推荐序

早在1842年，意大利数学家Luigi Menabrea 就在其《查尔斯·巴贝奇的分析引擎》（《Sketch of the Analytical Engine Invented by Charles Babbage》）一文中阐述了并行的理念。他的这篇文章涉及计算机系统结构和程序设计的诸多方面。到了1958年，S.Gill针对并行程序设计进行了探讨，而IBM的John Cocke和Daniel Slotnick则针对并行数值计算进行了探讨。然而直到1962年，第一台多处理计算机才藉由Burroughs公司的D825而面世。此后，为数众多的公司提出并推出了不同系统结构的多处理器系统。

曾几何时，工业界和学术界进行了无数次的尝试，力图将并行程序设计置于主流程序设计的核心地位，但由于种种原因，这些尝试最终都未能取得成功。最主要的原因在于我们既然有能力将处理器的主频每12～18个月就增加一倍，因此除了那些依赖于大规模并行系统来加速其模拟运算的科学社团外，其他人就几乎找不到任何理由来编写并行程序。

然而近来，半导体工业界与微处理器供应商却遭遇到了屏障，即虽然我们依然能增加芯片的密度，但却再也无法增加主频。有鉴于此，多核处理器开始登场，比如Sun的Niagara系列，IBM的Power系列，AMD的Opteron，以及Intel的Xeon。事实上，供应商们未来将会提供时钟速度更低但核数更多的处理器，如Intel的Nehalem以及AMD的Istanbul。此外，其他形式的加速器也纷纷登场，例如GPGPU（如Nvidia的Tesla，AMD的Firestream，Intel的Larabee），Cell处理器，以及FPGA。

这就意味着，为了能充分利用多核处理器，我们必须寻找应用内在的并行性，必须要编写具有并行线程的代码。我们已然预见到串行程序会“逐渐减速”，以此观之，似乎并行程序设计的概念终将成为未来主流程序员所必须掌握的重要概念。

《并行程序设计原理》一书对学生以及初学者将颇有裨益。两位作者Calvin Lin和Lawrence Snyder非常细致地将各种原则与当前的并行硬件和软件场景联系了起来，使得本书既能扎根于计算机科学的基础，又能与时俱进。这本全新的可实践且实用的书，必将使得阅读引人入胜且充满趣味。

本书着力介绍一系列不同类型的程序设计工具：如在共享存储器程序设计中介绍了Pthread、Java Threads以及OpenMP；而在消息传递程序设计中将MPI作为局部视图语言的代表，并提及了UPC等PGAS语言，而将ZPL作为全局视图语言的典范；此外还提及了3个最新的并行程序设计语言，即Sun的Fortress，IBM的X10，以及Cray的Chapel。

本书还探讨了一些高级内容：比如“理想”的并行程序设计语言所应遵循的基本原则；并行软件开发的方法学和方式，如在敏捷软件开发中最常用的增量式开发。

本书无论对于积极进取的HPC业内人士来说，还是通用程序员而言，都是一本极具价值的参考书。总之，这本可靠而翔实的书探讨了整个社团所将面临的有关并行化的挑战与问题。

Simon See博士
Sun高性能计算及云计算技术中心主任兼首席科学家
新加坡南洋理工大学兼职副教授
2009年6月

译者序

随着多核体系结构的出现和快速发展，使得并行计算科学的硬件基础设施发生了很大变化，如果把并行硬件基础设施看成是“经济基础”，则其相应的上层并行软件就可以视为“上层建筑”。由于“经济基础”的变化，作为其中重要的“上层建筑”之一的并行程序设计技术，必须进行相应的变化以适应新的“经济基础”。因此如何在多核体系结构上进行高效的并行程序设计以充分利用多核所提供的硬件并行性，从而大幅度地提升并行计算性能指标就显得非常重要。本书的主要论题之一便是环绕这一问题展开的。

并行程序设计比顺序程序设计要困难得多，一是因为并行程序设计的平台不是唯一的，存在多种不同的并行体系结构，而顺序程序设计只有唯一的冯·诺依曼体系结构；二是因为并行程序有多个进程或线程在同时运行，它们之间往往需要进行通信和同步，这就使并行程序设计变得复杂，特别是在要获得线性加速比时尤为如此；三是因为并行程序设计没有顺序程序设计中如C及Java那样通用和普及的并行程序设计语言，因此只能针对不同的体系结构选择使用不同的并行语言或例程库，例如对于共享地址空间的体系结构就必须选择如OpenMP、Java Threads或POSIX Threads那样的语言，而对于分布地址空间的体系结构就不得不选择如MPI或PVM那样的例程库语言。此外若要开发数据并行性，则就需要选用高性能的如Fortran（HPF）那样的语言。本书另一个主要论题便是环绕这一问题展开的。

本书侧重论述并行程序设计的基本原理，解释各种现象，并分析为何这些现象意味着成功进行并行程序设计的机遇或是阻碍。并行的硬件基础设施和并行的软件设计环境随着时间的变迁会不断发生变化，但原理则永远不会过时。以原理作为第一要素进行论述是本书的特色之一。

本书的另一个特色是，它侧重可扩展性和可移植性，即所设计的并行程序具有在任何数目处理器系统上和在任何并行体系结构平台上运行良好的能力。这一概念在多核时代是非常关键的，这是因为：首先，使得并行计算具有可扩展能力的大多数技术与在多核芯片上生成高效求解的技术是相同的；其次，虽然目前的多核芯片所具有的处理器数目还比较小，通常是2～8个，但今后每个芯片上的核数将会急剧增加，这就使得可扩展并行概念与之直接相关；最后，显然我们应该侧重研究和开发那些在现在和将来都能很好工作的方法。

内容非常实用是本书的又一特色。这是因为本书在介绍并行程序设计系统的同时，还叙述如何在这些系统中应用并行程序的设计原理。作者通过自身丰富的实践经验为读者介绍了在从事并行程序设计时应遵循的方法学。译者认为从事并行程序设计者应注重对并行程序设计方法学的了解、掌握，以及有关素质的培养，唯此才能开发出性能良好以及生命力持久的并行程序，并提高编制并行程序的能力和生产率。

翻译本书的原因有两个：一是本书的内容相当新，涉及现代的并行硬件和软件技术，包括多核体系结构及其并行程序设计技术；二是本书论述了并行程序设计中的一些深层次问题，如可扩展性、可移植性以及并行程序设计应遵循的方法学等。本书的不足之处在于对一些性能问题的定量分析不够充实，此外所介绍的并行机抽象模型也不够全面，只有一个CTA模型。但这些瑕疵并不会影响本书的阅读价值。

本书是计算机科学专业本科高年级学生或一年级硕士生的理想教科书，对专业程序员来讲则是从事并行程序设计的一本理想入门书。本书对软件工程师和计算机系统设计师也是非常值得一读的参考书。

本书的翻译工作由陆鑫达教授负责和组织。陆鑫达教授翻译了目录、前言、第1~4章以及第10~11 章，林新华老师翻译了第5~9章。译稿全文由陆鑫达教授统稿审校。原书只分章不分节，为方便读者阅读，我们统一进行了分节。值此中译本出版之际，译者特向机械工业出版社华章公司的策划和编辑人员表示深切的谢意。

书中的术语翻译，我们尽量采用已公布的计算机科学技术名词（第二版），对于一些未公布的术语（包括一些新出现的术语）我们尽量采用流行的译法。由于时间较为仓促，翻译中的错误或不妥之处在所难免，敬请广大读者不吝指正。

致谢

在本书行将付梓之前，承蒙Sun公司高性能计算及云计算技术中心主任兼首席科学家Simon See博士拨冗为本译著撰写了推荐序，在此仅表深切的谢意！

上海交通大学计算机科学与工程系
陆鑫达　林新华
2009年6月10日

前　言

对于那些因多核芯片出现而激发起学习并行程序热情的读者来说，你找对了地方。本书是为并行计算机无处不在的这个世界而编写的，这种并行计算机涉及广泛，从具有两核芯片的膝上计算机，到超级计算机，再到用于搜索因特网的巨大数据中心机群。

本书侧重可扩展并行，即并行程序具有在任何数目处理器上运行良好的能力。这一概念是非常关键的，原因有二：（1）创建可扩展并行计算所需的大多数技术与在多核芯片上生成高效求解的技术是相同的；（2）目前的多核芯片具有适中数目的处理器，通常是2～8个，在未来几年内每个芯片上的核数肯定会急剧增加，这就使得可扩展并行概念与之直接相关。因而，尽管今天的多核芯片为核间低时延通信提供了机遇，但这种特性很可能只能在短期内得益，因为当芯片上的核数增长时，芯片内不同部分的片内延迟将明显增加。所以，我们不会侧重开发这种短期得益的方法，而是侧重那些能在现在和将来都能很好工作的方法。当然，多核芯片还面临着自身的挑战，特别是它们有限的片外存储器带宽，以及有限的片上总的cache容量。本书也将讨论这些问题。

首先，我们讨论构成实用和高效并行程序的原理。为了获取如程序设计那样复杂的能力，学习原理是至关重要的。当然，对于并行程序设计来讲，原理也许更加重要，因为最新技术水平的变化很快。如果训练过于紧密地捆绑于某个特定的计算机系统或一种语言，将不具有跟上技术发展步伐的支撑力。可是原理（应用于任何并行计算系统的概念以及开发这些特征的观念）对人们深入理解和掌握相关知识将提供深远的影响。

但我们并不会止步于对抽象概念的讨论，还会将那些原理应用到日常的计算中，这将使本书非常实用。我们将介绍几个并行程序设计系统，还将描述在程序设计系统中如何应用这些原理。我们期望读者在读完本书后能编写并行程序。在最后一章我们将专门讨论并行程序设计技术，以及介绍如何开发一个有关并行程序设计的结课课程设计，这个设计可能需要历时一个学期。

读者对象

读者对象可以是任何人，大学生或专业人员，只要他能用C或类似语言成功进行编程，以及自认为是程序员的人。如果具有计算机执行顺序程序的概念，包括取指/执行周期以及高速缓存基础的知识，则就更易理解本书的内容。本书最初的定位是计算机科学专业的高年级学生以及具有计算机科学学士学位的一年级研究生，现在本书仍然适合这一程度。然而，作为本书的宗旨，我们减少了对预备知识的要求，而强调对知识的传授，如果读者已经掌握某些解释所涉及的知识，只需跳过这些知识。

本书结构

因为并行程序设计并不是读者所熟悉的顺序程序设计的直接扩展，所以我们将本书分为四部分：

基础：第1～3章

并行抽象：第4～5章

并行程序设计语言：第6～9章

展望：第10～11章

为使读者能明智地从中选择感兴趣的部分，我们下面说明各部分的目的和内容。

基础 在第1章中，我们通过实现一个计算说明并行程序员必须处理的许多问题是多么困难，而在为顺序计算机编写同样的程序时则非常简单。该例子将我们的注意力集中到一些贯穿于全书与我们相关的论题，但它也强调了解并行计算机是如何操作的重要性。第2章介绍5种不同类型的并行计算机，描述了其体系结构中的少量细节，以及它们扩展到更大规模的能力。在这一章中得出了两个关键的结论：第一，不像顺序计算，并行计算机没有一个标准的体系结构。第二，为了能适应这种体系结构的多样性，我们需要一个抽象的机器模型以指导我们的程序设计。我们给出了一个抽象模型。在留意体系结构的基础上，第3章的内容涉及并发性（包括线程和进程）、时延、带宽、加速比等基本概念，侧重叙述与性能相关的问题。第一部分这些基础内容为第二部分讲解算法和抽象做好了准备。

并行抽象 为支持并行算法的设计和讨论，第4章为编写独立于语言的并行程序引入了一个非正式的伪代码符号。该符号具有跨各种程序设计模型和方法的许多特性，允许我们在讨论算法时不会偏向任何具体的语言或机器。为了引导读者考虑并行算法，第5章介绍了一系列的基本算法技术。在阅读完第二部分后，读者将掌握以并行方式求解问题的方法，从而可使我们进入最后一个问题，即以一个具体的并行程序设计语言来编码算法。

并行程序设计语言 还没有一种并行程序设计语言，能够如C或Java在顺序程序设计中所起的作用那样，为大家所熟知和接受，并作为编码算法的基本手段。为此，第三部分介绍了三种并行程序设计语言：基于线程的语言（第6章），消息传递语言（第7章）和高级语言（第8章）。我们所论及的每种语言的内容，足以使读者能编写小的练习；但编写重大的问题将需要更完整的语言介绍，这可通过在线资源获得。除了介绍一种语言，每一章还包含一个有关语言的简短综述，这些语言在并行程序设计社团中有一批追随者。第9章简单地比较和对比了所有已论述的语言，指明它们的优缺点。能通读这三章最好，但是我们知道许多读者将只会定格在一种方法上，因此这三章是相互独立的。

展望 第四部分是展望未来。第10章涉及一系列新的、有前途的并行技术，它们无疑将影响未来的研究和实践。我们的观点是，这些技术还未为它们的“全盛时期做好准备”，但它们是重要的，且值得我们去熟悉它们，即使在它们还未被完全部署之前。最后，第11章侧重论述并行机程序设计的第一手实践技术。该章的前两节应在研究并行程序设计之前阅读，也许与第4章和第5章的抽象一起学习更好。但该章的主要目的是帮助读者编写作为结课课程设计内容的一个真实程序。

本书使用方法

虽然本书的内容以逻辑顺序排列，但也不必从头到尾阅读本书。的确，作为一个学期的课程，在学完所有论题之前就开始程序设计的练习，这可能是一个明智的选择。请看如下的一个明智的学习计划：

- 第1章和第2章
- 11.1节，3.4节；开始程序设计练习
- 第4章和第5章

- 第6～8章（关于程序设计语言）之一
- 完成第3章和第11章，然后开始学期课程设计
- 依次完成以下各章：关于语言的各章，第9章和第10章

当然，从头到尾阅读本书并无坏处，但上述方法的优点是阅读和程序设计可并行进行。

致谢

要真诚地感谢E Christopher Lewis 和Robert van de Geijn两位对本书初稿的评论。也要对以下评阅人的宝贵反馈意见和建议表示感谢：

David Bader, 佐治亚技术学院

Purushotham Bangalore, 亚拉巴马大学伯明翰分校

John Cavazos, 特拉华大学

Sandhya Dwarkadas, 罗切斯特大学

John Gilbert, 加州大学圣巴巴拉分校

Robert Henry, Cray公司

E Christopher Lewis, VMWare

Kai Li, 普林斯顿大学

Glenn Reinman, UCLA（加州大学洛杉矶分校）

Darko Stefanovic, 新墨西哥大学

我们感谢Karthik Murthy和 Brandon Plost两位在编写和运行并行程序中的帮助，以及帮助发现正文中的瑕疵，我们还要感谢Bobby Blumofe, 他与我们在多线程程序设计课程中的早期合作在本书的许多地方都可看到。我们赞赏和感谢华盛顿大学2006年秋季学期并行程序设计环境讨论班（CSE590o）的学生们对本教科书的贡献，他们是：Ivan Beschastnikh，Alex Colburn，Roxana Geambasu，Sangyun Hahn，Ethan Katz Bassett, Nathan Kuchta, Harsha Madhyastha, Marianne Shaw, Brian Van Essen, Benjamin Ylvisaker。Sonja Keserovic, Kate Moore, Brad Chamberlain，Steven Deitz, Dan Grossman, Jeff Diamond, Don Fussell, Bill Mark和David Mohr。

我们要向编辑Matt Goldstein以及Addison Wesley出版社团队成员Sarah Milmore、Marilyn Lloyd、Barbara Atkinson、Joyce Wells和Chris Kelly表示感谢。谢谢Gillian Hall 对我们愚钝行为的特别容忍。

最后，我们要感谢我们的家人在撰写本书期间对我们的容忍。

Calvin Lin

Lawrence Snyder

2008年2月

目　录

第二部分 并行抽象

第三部分 并行程序设计语言

第四部分 展 望

第一部分 基 础

学习并行程序设计先要打好结实的基础。基础中最重要的就是要能分辨顺序程序设计和并行程序设计两者的差异。在顺序计算中，每次只能完成一个操作，使得判断一个程序的正确性和它的性能特性非常直截了当。在并行计算中，许多操作同时进行，使得对程序的正确性和性能的判断变得复杂，因此需要修改我们的程序设计方法。本篇将阐明这一差异所带来的主要后果。

第1章是并行计算的导论，从求解在一维数组中统计3出现次数的一个简单问题开始。这个任务看似普通，在创建一个有合理性能的程序之前，却需要进行四次尝试。不仅如此，我们发现对加速比的最大期望无法实现。在讲解这一例子时，我们将介绍许多并行的基本概念。

第2章描述并行计算机的基本体系结构特征。就其本身而言，便是一个饶有兴趣的主题，因为像处理器间通信这样富有挑战性的问题存在许多的可能解答，而且体系结构师所使用的技术具有很强的独创性。对各种并行机进行考察得出的结论是它们是截然不同的。由于程序员需要知晓机器低层的一些特性才能编写高质量的程序，因此有必要寻找一个能统一各种不同体系结构的机器模型。我们将引入这样的一个模型作为后面研究的基础。

为了清晰地了解并行计算机是如何工作的，在第3章中从概念性角度剖析了许多有关并行性能的问题。我们介绍的主要概念包括时延、带宽、加速比和效率。在程序设计方面，如其中的相关性，将作为并行线程的干扰源进行重点介绍。一旦掌握了所介绍的这些基本概念，就为学习第二部分中的算法思想做好了准备。

第1章 导 论

并行计算是一种能加速计算的基本技术，因此不断增长的并行硬件可用性隐含着巨大的机遇。但是实现一个并行求解意味着对某些概念和程序设计的挑战，这正是本书试图要解决的。为了正确地权衡机遇和挑战，本章将提出与并行程序有关的问题并介绍基本概念。

1.1 并行的威力和潜能

并行在日常生活中时常碰到。更为重要的是，在过去的几十年中并行以各种方式为计算机性能的稳定提高做出了贡献。现在新的机遇正在出现。下面让我们来详细论述这一点。

1.1.1 并行，一个熟悉的概念

并行是我们熟悉的一个概念。杂耍（juggling）是人类能完成的一个并行任务。房屋建造是一种并行活动，因为几个工人能同时完成不同的工作，如电线配线、水管安置、锅炉管道安装等等。大多数的工业产品如汽车、吹风器、速冻食品，都以流水线方式进行生产，在流水线上正在建造的许多单件产品是同时进行加工或装配的。呼叫中心则是另一种应用并行的结构，其中有许多雇员在同时为顾客服务。

虽然熟悉，但应注意这些并行形式是不同的。例如呼叫中心在本质上与房屋建造有所不同：呼叫通常是独立的，因此能以任意次序提供服务，而且工作人员之间几乎没有交互。而在建房时，某些任务能同时完成，如电线配线和水管安置，而另外一些任务则必须依次进行，例如配线架必须在配线之前进行安装。这种顺序限制了能同时进行的并行量，从而也就限制了一个建设项目完成的进度。顺序性也增加了工人之间的交互程度。制造业的流水线与前两者又有所不同，因为它们有严格的顺序约束，制造产品的各个阶段通常以顺序方式进行；并行性来自于流水线上同时在生产许多单件产品。杂耍则属于事件驱动的并行，当一个事件（一个落下的球）发生时将引起有关操作的执行（抓球、抛球）以响应该事件。这些熟悉的并行形式也将出现在我们要讨论的并行计算中。

1.1.2 计算机程序中的并行

以并行方式执行程序指令的主要动机是为了加快计算。但是现今的大多数程序是无法通过并行来改进性能的，因为它们的编写是假设指令是按序执行的，每次执行一条指令，即是顺序执行的。大多数程序设计语言的语义体现为顺序执行，按这种语义编写的程序通常严重地依赖于这种特征来保证正确性，因而很难有机会实现并行执行。当然，还存在一些机会，例如在计算表达式(a+b)*(c+d)时，假定这些都是简单变量，由于子表达式(a+b)和(c+d)相互独立，因此它们能同时计算。这就是指令级并行（Insruction Level Parallelism，ILP）的例子。

的确，我们继续编写顺序程序的一个理由是因为计算机体系结构师们在开发并行性方面如此成功。他们利用硅工艺的稳定改进，在顺序处理器的设计中采用了多种并行方法，其中包括ILP。首先，体系结构师们为指令和数据提供了分离的路径和高速缓存（cache）。这种分

离允许对指令和数据存储器进行并行访问，两者互不干扰。其次，使指令执行流水化，在执行当前指令的同时，对后继的指令进行取指和译码，而前面的指令则正在将结果写回存储器。更进一步，处理器每次可发出（启动）多条指令，预取指令和数据，以猜测方式并行地完成操作，即使不能保证是否真的需要进行这些操作，他们还使用高速并行电路完成基本的算术操作。简言之，现代处理器是高度并行的系统。

对程序员来讲，所有这些并行对顺序程序都是透明可用的。我们称这种并行为*隐藏并行*(hidden parallelism)。这种并行，加上不断提高的时钟速度，使得新一代的处理器芯片总能以更快的速度执行程序，而同时又保持顺序执行的假象。但是要在维持顺序语义的基础上期望找到应用并行的新机遇是非常有限的。更为严重的是，就功率消耗和性能两者而言，开发ILP的现有技术在很大程度上已经到达了它的收益递减点。因而，就当前的技术而言，顺序程序的执行可能已接近它的最大速度。

要想继续获取显著的性能改进，必须超越现有程序典型的单指令序列的工作方式。我们必须使程序以同时运行多指令流的方式工作。这种工作方式需要使用新的程序设计技术，这就是本书要讨论的主题。

1.1.3 多核计算机，一个机遇

虽然单处理器的性能改进可能已接近极限，但摩尔定律的预言使得晶体管密度仍在不断提高。芯片制造商利用这一机会，在单个芯片上放置多个指令执行引擎以及相应的高速缓存。这种结构很快得到新的名称“核”(core)，因为它代表了一个典型顺序处理器的核心部分。早期的多核芯片有2个、4个或8个核，但是核的数目随着新一代处理器的出现在不断增长。

于2005/2006年问世的第一个多核芯片引起了全社会关于“免费午餐结束”的讨论。讨论的焦点如下：

- 由于如前所述的硅工艺和体系结构设计（隐藏并行）的进步，软件开发者几十年来一直享受稳定的性能改进——“免费午餐”。
- 由于无须关心性能，多年来程序员对他们使用的技术和方法学几乎没有什么改变（面向对象的范例是一个值得一提的例外）。
- 现有的软件通常不能直接利用多核芯片。
- 不能利用多核芯片的程序在目前和未来都无法实现性能改进。
- 大多数程序员不知道如何编写并行程序。

令人不快的结论是程序需要改变，为此程序员也必须随之改变。

虽然上述的结论在某些人看来可能是一个坏消息，但同时还有相应的好消息。特别是，如果将一个计算重写成并行形式，而且如果该并行程序是*可扩展的*（表示它能够使用日益增多的处理器），那么随着硅工艺的进步，随之会有更多的核加到未来的芯片上，则已重写的程序的性能将持续上升。但对不可扩展的并行程序来讲，将不能享受硅工艺进步的好处。因此，获取可扩展并行至关重要。

某些特别是在图形学领域的研究人员，无疑对是否需要并行计算的讨论感到非常奇怪，因为他们多年来一直在使用并行性。图形处理部件（Graphics Processing Unit，GPU），又称为图形卡，已经是加速绘制（rendering）流水线的标准技术。虽然GPU看起来像是一个小巧的协处理器，对通用计算机应用来讲微不足道，但硅工艺的进展已使得*在图形处理部件上进行通用计算*（General Purpose computing on Graphics Processing Unit，GPGPU）这一概念成