第一图书网, tushu007.com

<<并行程序设计导论>>

图书基本信息

书名：<<并行程序设计导论>>

13位ISBN编号：9787111392842

10位ISBN编号：7111392841

出版时间：2012-12-1

出版时间：机械工业出版社华章公司

作者：Peter Pacheco

页数：252

译者：邓倩妮

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：http://www.tushu007.com

内容概要

《并行程序设计导论/计算机科学丛书》编著者Peter S.Pacheco。

本书全面涵盖了并行软件和硬件的方方面面，深入浅出地介绍如何使用MPI（分布式内存编程）、Pthreads和OpenMP（共享内存编程）编写高效的并行程序。
各章节包含了难易程度不同的编程习题。
本书可以用做计算机科学专业低年级本科生的专业课程的教材，也可以作为软件开发人员学习并行程序设计的专业参考书。

作者简介

作者：（美国）帕切克（Peter S.Pacheco） 邓倩妮 Peter S.Pacheco拥有佛罗里达州立大学数学专业博士学位。
曾担任旧金山大学计算机系主任，目前是旧金山大学数学系主任。
近20年来，一直为本科生和研究生讲授并行计算课程。

第一图书网, tushu007.com

<<并行程序设计导论>>

书籍目录

出版者的话
译者序
本书赞誉
前言
致谢
第1章 为什么要并行计算
1.1 为什么需要不断提升的性能
1.2 为什么需要构建并行系统
1.3 为什么需要编写并行程序
1.4 怎样编写并行程序
1.5 我们将做什么
1.6 并发、并行、分布式
1.7 本书的其余部分
1.8 警告
1.9 字体约定
1.10 小结
1.11 习题
第2章 并行硬件和并行软件
2.1 背景知识
2.1.1 冯·诺依曼结构
2.1.2 进程、多任务及线程
2.2 对冯·诺依曼模型的改进
2.2.1 Cache基础知识
2.2.2 Cache映射
2.2.3 Cache和程序:一个实例
2.2.4 虚拟存储器
2.2.5 指令级并行
2.2.6 硬件多线程
2.3 并行硬件
2.3.1 SIMD系统
2.3.2 MIMD系统
2.3.3 互连网络
2.3.4 Cache一致性
2.3.5 共享内存与分布式内存
2.4 并行软件
2.4.1 注意事项
2.4.2 进程或线程的协调
2.4.3 共享内存
2.4.4 分布式内存
2.4.5 混合系统编程
2.5 输入和输出
2.6 性能
2.6.1 加速比和效率
2.6.2 阿姆达尔定律
2.6.3 可扩展性

2.6.4 计时
2.7 并行程序设计
2.8 编写和运行并行程序
2.9 假设
2.10 小结
2.10.1 串行系统
2.10.2 并行硬件
2.10.3 并行软件
2.10.4 输入和输出
2.10.5 性能
2.10.6 并行程序设计
2.10.7 假设
2.11 习题
第3章 用MPI进行分布式内存编程
3.1 预备知识
3.1.1 编译与执行
3.1.2 MPI程序
3.1.3 MPI_Init和MPI_Finalize
3.1.4 通信子、MPI_Comm_size和MPI_Comm_rank
3.1.5 SPMD程序
3.1.6 通信
3.1.7 MPI_Send
3.1.8 MPI_Recv
3.1.9 消息匹配
3.1.10 status_p参数
3.1.11 MPI_Send和MPI_Recv的语义
3.1.12 潜在的陷阱
3.2 用MPI来实现梯形积分法
3.2.1 梯形积分法
3.2.2 并行化梯形积分法
3.3 I/O处理
3.3.1 输出
3.3.2 输入
3.4 集合通信
3.4.1 树形结构通信
3.4.2 MPI_Reduce
3.4.3 集合通信与点对点通信
3.4.4 MPI_Allreduce
3.4.5 广播
3.4.6 数据分发
3.4.7 散射
3.4.8 聚集
3.4.9 全局聚集
3.5 MPI的派生数据类型
3.6 MPI程序的性能评估
3.6.1 计时
3.6.2 结果

3.6.3 加速比和效率
3.6.4 可扩展性
3.7 并行排序算法
3.7.1 简单的串行排序算法
3.7.2 并行奇偶交换排序
3.7.3 MPI程序的安全性
3.7.4 并行奇偶交换排序算法的重要内容
3.8 小结
3.9 习题
3.10 编程作业
第4章 用Pthreads进行共享内存编程
4.1 进程、线程和Pthreads
4.2 “ Hello， World ” 程序
4.2.1 执行
4.2.2 准备工作
4.2.3 启动线程
4.2.4 运行线程
4.2.5 停止线程
4.2.6 错误检查
4.2.7 启动线程的其他方法
4.3 矩阵-向量乘法
4.4 临界区
4.5 忙等待
4.6 互斥量
4.7 生产者-消费者同步和信号量
4.8 路障和条件变量
4.8.1 忙等待和互斥量
4.8.2 信号量
4.8.3 条件变量
4.8.4 Pthreads路障
4.9 读写锁
4.9.1 链表函数
4.9.2 多线程链表
4.9.3 Pthreads读写锁
4.9.4 不同实现方案的性能
4.9.5 实现读写锁
4.10 缓存、缓存一致性和伪共享
4.11 线程安全性
4.12 小结
4.13 习题
4.14 编程作业
第5章 用OpenMP进行共享内存编程
5.1 预备知识
5.1.1 编译和运行OpenMP程序
5.1.2 程序
5.1.3 错误检查
5.2 梯形积分法

5.3 变量的作用域
5.4 归约子句
5.5 parallelfor指令
5.5.1 警告
5.5.2 数据依赖性
5.5.3 寻找循环依赖
5.5.4 π值估计
5.5.5 关于作用域的更多问题
5.6 更多关于OpenMP的循环：排序
5.6.1 冒泡排序
5.6.2 奇偶变换排序
5.7 循环调度
5.7.1 schedule子句
5.7.2 static调度类型
5.7.3 dynamic和guided调度类型
5.7.4 runtime调度类型
5.7.5 调度选择
5.8 生产者和消费者问题
5.8.1 队列
5.8.2 消息传递
5.8.3 发送消息
5.8.4 接收消息
5.8.5 终止检测
5.8.6 启动
5.8.7 atomic指令
5.8.8 临界区和锁
5.8.9 在消息传递程序中使用锁
5.8.10 critical指令、atomic指令、锁的比较
5.8.11 经验
5.9 缓存、缓存一致性、伪共享
5.10 线程安全性
5.11 小结
5.12 习题
5.13 编程作业
第6章 并行程序开发
6.1 n体问题的两种解决方法
6.1.1 问题
6.1.2 两个串行程序
6.1.3 并行化n体算法
6.1.4 关于I/O
6.1.5 用OpenMP并行化基本算法
6.1.6 用OpenMP并行化简化算法
6.1.7 评估OpenMP程序
6.1.8 用Pthreads并行化算法
6.1.9 用MPI并行化基本算法
6.1.10 用MPI并行化简化算法
6.1.11 MPI程序的性能

6.2 树形搜索
6.2.1 递归的深度优先搜索
6.2.2 非递归的深度优先搜索
6.2.3 串行实现所用的数据结构
6.2.4 串行实现的性能
6.2.5 树形搜索的并行化
6.2.6 采用Pthreads实现的静态并行化树搜索
6.2.7 采用Pthreads实现的动态并行化树搜索
6.2.8 Pthreads树搜索程序的评估
6.2.9 采用OpenMp实现的并行化树搜索程序
6.2.10 OpenMp实现的性能
6.2.11采用MPI和静态划分来实现树搜索
6.2.12 采用MPI和动态划分来实现树搜索
6.3 忠告
6.4 选择哪个API
6.5 小结
6.5.1 Pthreads和OpenMP
6.5.2 MPI
6.6 习题
6.7 编程作业
第7章 接下来的学习方向
参考文献
索引

章节摘录

版权页： 插图： 接下来，我们仔细研究最后两个参数，由pthread_create生成并运行的函数应该有一个类似于下面函数的原型：void*.thread_function（void* arga_p）；因为类型voi d*可以转换为C语言中任意指针类型，所以args_P可以指向一个列表，该列表包含一个或多个thread_function函数需要的数值。
类似地，thread_function返回的值也可以是一个包含一个或多个值的列表。
在我们的代码中，调用pthread_create函数时，传入最后一个参数采用了一个常用的技巧：为每一个线程赋予了唯一的int型参数rank，表示线程的编号。
首先，我们先解释一下这么做的理由，然后再具体探讨如何做。
考虑以下问题：运行一个生成了两个线程的Pthreads程序，当其中一个线程遇到了错误时，我们或者用户如何才能知道是哪个线程出了问题呢？
我们不能简单地输出pthread_t对象，因为它是不透明的。
如果我们启动线程时赋予第一个线程编号为0，第二个线程编号为1，那么通过错误信息中线程的编号就能非常容易地判断是哪个线程出错了。
既然线程函数可以接收voi d*类型的参数，我们就可以在main函数中为每个线程分配一个int类型的整数，并为这些整数赋予不同的数值。
当启动线程时，把指向该int型参数的指针传递给pthread_create函数。
然而，程序员会用类型转换来处理此问题：不是在main函数中生成int型的进程号，而是把循环变量thread转化为voi d*类型，然后在线程函数hello中，把这个参数的类型转换为long型（第33行）。
类型转换的结果是“系统定义”的，但大多数C编译器允许这么做。
不过，如果指针类型的大小和表示进程编号的整数类型不同，在编译时就会收到警告。
在我们使用的机器上，指针类型是64位，而int型是32位，为了避免警告，我们用long型替代了int型。
需要注意的是，我们为每一个线程分配不同的编号只是为了方便使用。
事实上，pthread_create创建线程时没有要求必须传递线程号，也没有要求必须要分配线程号给一个线程。
还需要注意的是，并非由于技术上的原因而规定每个线程都要运行同样的函数。
一个线程运行hello函数的同时，另一个线程可以运行goodbye函数。
但与编写MPI程序的方法类似，Pthreads程序也采用“单程序，多数据”的并行模式，即每个线程都执行同样的线程函数，但可以在线程内用条件转移来获得不同线程有不同功能的效果。
4.2.4运行线程 运行main函数的线程一般称为主线程。
所以，在线程启动后，会打印一句：Hello from the main thread 同时，调用pthread_create所生成的线程也在运行。
这些线程通过第33行的类型转换代码获得各自的编号，然后打印各自的消息。
注意，当线程结束时，由于它的函数的类型有一个返回值，那么线程就应该返回一个值。
在本例中，线程没有需要特别返回的值，所以只返回NULL。
在Pthreads中，程序员不直接控制线程在哪个核上运行。
在pthread_create函数中，没有参数用于指定在哪个核上运行线程。
线程的调度是由操作系统来控制的。
在负载很重的系统上，所有线程可能都运行在同一个核上。
事实上，如果线程个数大于核的个数，就会出现多个线程运行在一个核上。
当然，如果某个核处于空闲状态，操作系统就会将一个新线程分配给这个核。

编辑推荐

《并行程序设计导论》是一本精心撰写的、全面介绍并行计算的书籍。
作者循序渐进地展示了如何利用MPI、Pthreads和OpenMP开发高效的并行程序，教给那些专业知识有限、没有并行化经验的读者如何开发、调试分布式内存和共享内存的程序，以及对程序进行性能评估。

名人推荐

毫无疑问，随着多核处理器和云计算系统的广泛应用，并行计算不再是计算世界中被束之高阁的偏门领域。
并行性已经成为有效利用资源的首要因素。
Peter Pachec0撰写的这本新教材对于初学者了解并行计算的艺术和实践很有帮助。
——Duncan Buell，南卡罗来纳大学计算机科学与工程系 本书阐述了两个越来越重要的领域：使用Plhreads和OpenMP进行共享内存编程。
以及使用MPl进行分布式内存编程。
更重要的是。
通过指出可能出现的性能错误，强调好的编程实现的重要性。
这些主题包含在计算机科学、物理学和数学等多个学科中。
各章包含了大量不同难易程度的编程习题。
对于希望学习并行编程技巧、更新知识面的学生或专业人士来说，本书是一本理想的参考书。
——Leigh Little，纽约州立大学布罗科波特学院计算机科学系

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:http://www.tushu007.com