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：9787111358282

10位ISBN编号：7111358287

出版时间：2011-9

出版时间：机械工业出版社

作者：（美）Peter S. Pacheco

页数：370

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

采用教程形式，从简短的编程实例起步，一步步编写更有挑战性的程序。
重点介绍分布式内存和共享式内存的程序设计、调试和性能评估。
使用MPI、PTrlread和OperIMP等编程模型，强调实际动手开发并行程序。

并行编程已不仅仅是面向专业技术人员的一门学科。
如果想要全面开发机群和多核处理器的计算能力，那么学习分布式内存和共享式内存的并行编程技术是不可或缺的。
由Peter
S.Pacheco编著的《并行程序设计导论(英文版)》循序渐进地展示了如何利用MPI、PThread和OperlMP开发高效的并行程序，教给读者如何开发、调试分布式内存和共享式内存的程序，以及对程序进行性能评估。

第一图书网, tushu007.com

作者简介

帕切克(Petm
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

CHAPTER 1 Why Parallel Computing?
1.1 Why We Need Ever-Increasing Performance
1.2 Why We're Building Parallel Systems
1.3 Why We Need to Write Parallel Programs
1.4 How Do We Write Parallel Programs?
1.5 What We'll Be Doing
1.6 Concurrent, Parallel, Distributed
1.7 The Rest of the Book
1.8 A Word of Warning
1.9 Typographical Conventions
1.10 Summary
1.11 Exercises
CHAPTER 2 Parallel Hardware and Parallel Software
2.1 Some Background
2.1.1 The von Neumann architecture
2.1.2 Processes, multitasking, and threads
2.2 Modifications to the von Neumann Model
2.2.1 The basics of caching
2.2.2 Cache mappings
2.2.3 Caches and programs: an example
2.2.4 Virtual memory
2.2.5 Instruction-level parallelism
2.2.6 Hardware multithreading.
2.3 Parallel Hardware
2.3.1 SIMD systems
2.3.2 MIMD systems 32
2.3.3 Interconnection networks
2.3.4 Cache coherence
2.3.5 Shared-memory versus distributed-memory
2.4 Parallel Software 47
2.4.1 Caveats 47
2.4.2 Coordinating the processes/threads
2.4.3 Shared-memory 49
2.4.4 Distributed-memory
2.4.5 Programming hybrid systems
2.5 Input and Output 56
2.6 Performance 58
2.6.1 Speedup and efficiency
2.6.2 Amdahl's law 61
2.6.3 Scalability
2.6.4 Taking timings
2.7 Parallel Program Design
2.7.1 An example
2.8 Writing and Running Parallel Programs
2.9 Assumptions

第一图书网, tushu007.com

2.10 Summary
2.10.1 Serial systems
2.10.2 Parallel hardware
2.10.3 Parallel software
2.10.4 Input and output
2.10.5 Performance.
2.10.6 Parallel program design
2.10.7 Assumptions
2.11 Exercises
CHAPTER 3 Distributed-Memory Programming with MPI
3.1 Getting Started 84
3.1.1 Compilation and execution
3.1.2 MPI programs
3.1.3 MPI Init and MPI Finalize
3.1.4 Communicators, MPI Comm size and MPI Comm rank
3.1.5 SPMD programs
3.1.6 Communication
3.1.7 MPI Send
3.1.8 MPI Recv
3.1.9 Message matching
3.1.10 The status p argument 92
3.1.11 Semantics of MPI Send and MPI Recv 93
3.1.12 Some potential pitfalls 94
3.2 The Trapezoidal Rule in MPI 94
3.2.1 The trapezoidal rule 94
3.2.2 Parallelizing the trapezoidal rule 96
Contents xiii
3.3 Dealing with I/O 97
3.3.1 Output 97
3.3.2 Input100
3.4 Collective Communication 101
3.4.1 Tree-structured communication 102
3.4.2 MPI Reduce 103
3.4.3 Collective vspoint-to-point communications 105
3.4.4 MPI Allreduce 106
3.4.5 Broadcast 106
3.4.6 Data distributions 109
3.4.7 Scatter 110
3.4.8 Gather 112
3.4.9 Allgather 113
3.5 MPI Derived Datatypes 116
3.6 Performance Evaluation of MPI Programs 119
3.6.1 Taking timings 119
3.6.2 Results 122
3.6.3 Speedup and efficiency 125
3.6.4 Scalability 126
3.7 A Parallel Sorting Algorithm 127

第一图书网, tushu007.com

3.7.1 Some simple serial sorting algorithms 127
3.7.2 Parallel odd-even transposition sort 129
3.7.3 Safety in MPI programs 132
3.7.4 Final details of parallel odd-even sort 134
3.8 Summary 136
3.9 Exercises 140
3.10 Programming Assignments .147
CHAPTER 4 Shared-Memory Programming with Pthreads .151
4.1 Processes, Threads, and Pthreads 151
4.2 Hello, World 153
4.2.1 Execution 153
4.2.2 Preliminaries 155
4.2.3 Starting the threads 156
4.2.4 Running the threads 157
4.2.5 Stopping the threads 158
4.2.6 Error checking 158
4.2.7 Other approaches to thread startup159
4.3 Matrix-Vector Multiplication 159
4.4 Critical Sections 162
xiv Contents
4.5 Busy-Waiting 165
4.6 Mutexes .168
4.7 Producer-Consumer Synchronization and Semaphores 171
4.8 Barriers and Condition Variables 176
4.8.1 Busy-waiting and a mutex 177
4.8.2 Semaphores 177
4.8.3 Condition variables 179
4.8.4 Pthreads barriers 181
4.9 Read-Write Locks 181
4.9.1 Linked list functions 181
4.9.2 A multi-threaded linked list 183
4.9.3 Pthreads read-write locks 187
4.9.4 Performance of the various implementations 188
4.9.5 Implementing read-write locks 190
4.10 Caches, Cache Coherence, and False Sharing 190
4.11 Thread-Safety 195
4.11.1 Incorrect programs can produce correct output 198
4.12 Summary 198
4.13 Exercises 200
4.14 Programming Assignments .206
CHAPTER 5 Shared-Memory Programming with OpenMP .209
5.1 Getting Started 210
5.1.1 Compiling and running OpenMP programs 211
5.1.2 The program 212
5.1.3 Error checking215
5.2 The Trapezoidal Rule 216
5.2.1 A first OpenMP version 216

第一图书网, tushu007.com

5.3 Scope of Variables 220
5.4 The Reduction Clause .221
5.5 The parallel for Directive 224
5.5.1 Caveats 225
5.5.2 Data dependences 227
5.5.3 Finding loop-carried dependences 228
5.5.4 Estimating 229
5.5.5 More on scope231
5.6 More About Loops in OpenMP: Sorting .232
5.6.1 Bubble sort 232
5.6.2 Odd-even transposition sort 233
5.7 Scheduling Loops 236
5.7.1 The schedule clause 237
5.7.3 The dynamic and guided schedule types 239
5.7.4 The runtime schedule type 239
5.7.5 Which schedule? 241
5.8 Producers and Consumers 241
5.8.1 Queues241
5.8.2 Message-passing 242
5.8.3 Sending messages 243
5.8.4 Receiving messages 243
5.8.5 Termination detection 244
5.8.6 Startup 244
5.8.7 The atomic directive 245
5.8.8 Critical sections and locks 246
5.8.9 Using locks in the message-passing program 248
5.8.10 critical directives, atomic directives, or locks? 249
5.8.11 Some caveats 249
5.9 Caches, Cache Coherence, and False Sharing 251
5.10 Thread-Safety 256
5.10.1 Incorrect programs can produce correct output 258
5.11 Summary 259
5.12 Exercises 263
5.13 Programming Assignments .267
CHAPTER 6 Parallel Program Development 271
6.1 Two n-Body Solvers 271
6.1.1 The problem 271
6.1.2 Two serial programs 273
6.1.3 Parallelizing the n-body solvers 277
6.1.4 A word about I/O 280
6.1.5 Parallelizing the basic solver using OpenMP 281
6.1.6 Parallelizing the reduced solver using OpenMP 284
6.1.7 Evaluating the OpenMP codes 288
6.1.8 Parallelizing the solvers using pthreads 289
6.1.9 Parallelizing the basic solver using MPI 290
6.1.10 Parallelizing the reduced solver using MPI 292
6.1.11 Performance of the MPI solvers 297

第一图书网, tushu007.com

6.2 Tree Search 299
6.2.1 Recursive depth-first search 302
6.2.2 Nonrecursive depth-first search 303
6.2.3 Data structures for the serial implementations 305
6.2.6 A static parallelization of tree search using pthreads 309
6.2.7 A dynamic parallelization of tree search using pthreads 310
6.2.8 Evaluating the pthreads tree-search programs 315
6.2.9 Parallelizing the tree-search programs using OpenMP 316
6.2.10 Performance of the OpenMP implementations 318
6.2.11 Implementation of tree search using MPI and static partitioning 319
6.2.12 Implementation of tree search using MPI and dynamic partitioning 327
6.3 A Word of Caution 335
6.4 Which API? 335
6.5 Summary 336
6.5.1 Pthreads and OpenMP 337
6.5.2 MPI 338
6.6 Exercises 341
6.7 Programming Assignments 350
CHAPTER 7 Where to Go from Here 353
References 357
Index 361

第一图书网, tushu007.com

章节摘录

版权页：插图：There are many possible algorithms for identifying which subtrees we assign to the processes or threads.For example，one，thread or process could run the last version of serial depth.first search until the stack stores one partial tour for each thread or process.Then it could assign one tour to each thread or process.The problem wim depth.first searchisthatweexpecta subtreewhoserootisdeeperinthetreetorequire less work than a subtree whose root is higher up in the tree，so we would probably get better load balance if we used something like breadth.first search to identify t11e subtrees.As the name suggests，breadth-first search searches as widely as possible in the treebefore goingdeeper.Soif,forexample，we CalTyout abreadth-first searchuntil we reach alevel ofthe tree that has at least th reftd-count or comm-sz nodes.we can men divide the nodes at this level among the threads or processes.See Exercise 6.1 8 for implementation details. The best tour data structure On a shared-memory system，the best tour data structure can be shared.In this setting。
 the Feasibl e function Call simply examine the data structure.However,updates to the best tour will cause a race condition，and we U need some sort of locking t0 prevent errors.Wle ’ 11 discuss this in more detail when we implement the parallel version.In the case of a distributed-memory system，there are a couple of choices that we need to make about the best tour.T11e simplest option would be to have the processes operate independently of each other until they have completed searching their sub-trees.In this setting.each process would store its own local best tour.111is local best tourwouldbeusedbytheprocessin Fea s{b1 e andupdatedbytheprocesseachtime it calls Update-best tour.

第一图书网, tushu007.com

媒体关注与评论

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 Pactleco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PThread和OperIMP进行共享式内存编程，以及使用MPl进行分布式内存编程。
更重要的是，它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。
这本书在不同学科（包括计算机科学、物理和数学等）背景下介绍以上话题，各章节包含了难易程度不同的编程习题。
对于希望学习并行编程技巧、扩展知识面的学生或专业人士来说，这是一本理想的参考书籍。
——Leigh Little纽约州立大学布罗科波特学院计算机科学系本书是一本精心撰写的全面介绍并行计算的书籍，学生以及相关领域从业者会从书中的相关最新信息中获益匪浅。
作者以通俗易懂的写作手法，结合各种有趣的实例使本书引人入胜。
在并行计算这个瞬息万变、不断发展的领域里，本书深入浅出、全面涵盖了并行软件和硬件的方方面面。
——Kathy J.Liszka阿克隆大学计算机科学系

第一图书网, tushu007.com

<<并行程序设计导论>>

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