



An Introduction to

PARALLEL PROGRAMMING

Peter S. Pacheco

MK
MORGAN KAUFMANN

Contents

Preface	xv
Acknowledgments	xviii
About the Author	xix
CHAPTER 1 Why Parallel Computing?.....	1
1.1 Why We Need Ever-Increasing Performance	2
1.2 Why We're Building Parallel Systems	3
1.3 Why We Need to Write Parallel Programs	3
1.4 How Do We Write Parallel Programs?	6
1.5 What We'll Be Doing	8
1.6 Concurrent, Parallel, Distributed	9
1.7 The Rest of the Book	10
1.8 A Word of Warning	10
1.9 Typographical Conventions	11
1.10 Summary	12
1.11 Exercises	12
CHAPTER 2 Parallel Hardware and Parallel Software.....	15
2.1 Some Background.....	15
2.1.1 The von Neumann architecture	15
2.1.2 Processes, multitasking, and threads	17
2.2 Modifications to the von Neumann Model	18
2.2.1 The basics of caching.....	19
2.2.2 Cache mappings	20
2.2.3 Caches and programs: an example	22
2.2.4 Virtual memory	23
2.2.5 Instruction-level parallelism	25
2.2.6 Hardware multithreading.....	28
2.3 Parallel Hardware	29
2.3.1 SIMD systems	29
2.3.2 MIMD systems	32
2.3.3 Interconnection networks	35
2.3.4 Cache coherence	43
2.3.5 Shared-memory versus distributed-memory	46
2.4 Parallel Software	47
2.4.1 Caveats	47
2.4.2 Coordinating the processes/threads.....	48
2.4.3 Shared-memory	49

2.4.4	Distributed-memory	53
2.4.5	Programming hybrid systems	56
2.5	Input and Output	56
2.6	Performance	58
2.6.1	Speedup and efficiency	58
2.6.2	Amdahl's law	61
2.6.3	Scalability	62
2.6.4	Taking timings	63
2.7	Parallel Program Design	65
2.7.1	An example	66
2.8	Writing and Running Parallel Programs	70
2.9	Assumptions	70
2.10	Summary	71
2.10.1	Serial systems	71
2.10.2	Parallel hardware	73
2.10.3	Parallel software	74
2.10.4	Input and output	75
2.10.5	Performance	75
2.10.6	Parallel program design	76
2.10.7	Assumptions	76
2.11	Exercises	77
CHAPTER 3	Distributed-Memory Programming with MPI	83
3.1	Getting Started.....	84
3.1.1	Compilation and execution.....	84
3.1.2	MPI programs.....	86
3.1.3	<code>MPI_Init</code> and <code>MPI_Finalize</code>	86
3.1.4	Communicators, <code>MPI_Comm_size</code> and <code>MPI_Comm_rank</code>	87
3.1.5	SPMD programs	88
3.1.6	Communication	88
3.1.7	<code>MPI_Send</code>	88
3.1.8	<code>MPI_Recv</code>	90
3.1.9	Message matching	91
3.1.10	The <code>status_p</code> argument.....	92
3.1.11	Semantics of <code>MPI_Send</code> and <code>MPI_Recv</code>	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

3.3	Dealing with I/O	97
3.3.1	Output	97
3.3.2	Input	100
3.4	Collective Communication.....	101
3.4.1	Tree-structured communication.....	102
3.4.2	MPI_Reduce	103
3.4.3	Collective vs. point-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
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 startup	159
4.3	Matrix-Vector Multiplication	159
4.4	Critical Sections	162

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 checking	215
5.2	The Trapezoidal Rule	216
5.2.1	A first OpenMP version	216
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 π	229
5.5.5	More on scope	231
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
5.7.2	The static schedule type.....	238

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	Queues.....	241
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 <i>n</i> -Body Solvers	271
6.1.1	The problem.....	271
6.1.2	Two serial programs	273
6.1.3	Parallelizing the <i>n</i> -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
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
6.2.4	Performance of the serial implementations	306
6.2.5	Parallelizing tree search	306

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	