

Contents

Contributors	xi
Editor and Author Biographical Sketches.....	xiii
Symbol or Phrase	xix

CHAPTER 1 Editors' Introduction and Road Map	1
1.1 Why This Book?.....	1
1.2 Chapter Introductions	3
1.2.1 Part One—For Instructors.....	3
1.2.2 Part Two—For Students	5
1.3 How to Find a Topic or Material for a Course.....	6
1.4 Invitation to Write for Volume 2.....	7

PART 1 FOR INSTRUCTORS

CHAPTER 2 Hands-on Parallelism with no Prerequisites and Little Time Using Scratch.....	11
2.1 Contexts for Application.....	12
2.2 Introduction to Scratch.....	13
2.3 Parallel Computing and Scratch.....	14
2.3.1 Parallelism and Communication for Clean Solutions.....	14
2.3.2 A Challenge of Parallelism: Race Conditions	17
2.3.3 Blocking and Nonblocking Commands	20
2.3.4 Shared and Private Variables.....	21
2.4 Conclusion	23
References.....	23

CHAPTER 3 Parallelism in Python for Novices	25
3.1 Introduction	25
3.2 Background	26
3.2.1 Target Audience of this Chapter	26
3.2.2 Goals for the Reader.....	26
3.2.3 Tools.....	26
3.3 Student Prerequisites.....	27
3.3.1 Motivating Examples Involving Parallelism Concepts.....	27
3.3.2 Python Programming	28
3.4 General Approach: Parallelism as a Medium.....	29
3.5 Course Materials.....	30
3.6 Processes.....	30
3.6.1 Spawning a Process.....	31

3.6.2	Spawning Multiple Processes	32
3.6.3	Spawning Multiple Processes Using Pool	34
3.6.4	Anonymous Processes	34
3.6.5	Specifying Process Names	36
3.6.6	Using a Lock to Control Printing	36
3.6.7	Digging Holes	38
3.7	Communication	40
3.7.1	Communicating Via a Queue	40
3.7.2	Extended Communication Via a Queue	41
3.7.3	The Join Method	42
3.7.4	Obtaining a Result from a Single Child	43
3.7.5	Using a Queue to Merge Multiple Child Process Results	45
3.7.6	Mergesort Using Process Spawning and Queue Objects	46
3.7.7	Sharing a Data Structure	47
3.8	Speedup	48
3.8.1	Timing the Summation of Random Numbers	48
3.8.2	Using Join to Time a Child Process	49
3.8.3	Comparing Parallel and Sequential Performance	49
3.9	Further Examples Using the Pool/map Paradigm	52
3.9.1	Monte Carlo Estimation of π	52
3.9.2	Integration by Riemann Sum	54
3.9.3	Mergesort	54
3.10	Conclusion	54
	References	57
ER 4	Modules for Introducing Threads	59
4.1	Introduction	59
4.2	Prime Counting	61
4.2.1	Using this Module	62
4.2.2	Sequential Code	62
4.2.3	Step-by-Step Parallelization	63
4.2.4	Followup Assignment	71
4.3	Mandelbrot	72
4.3.1	Using this Module	73
4.3.2	Sequential Program	74
4.3.3	Step-by-Step Parallelization with OpenMP	75
4.3.4	Parallelization Using Explicit Threads	81
	References	82

CHAPTER 5	Introducing Parallel and Distributed Computing Concepts in Digital Logic	83
5.1	Number Representation	85
5.2	Logic Gates	90
5.2.1	Fan-Out and Fan-In of Gates	90
5.2.2	Tristate Gates and Buses	95
5.3	Combinational Logic Synthesis and Analysis	97
5.3.1	Timing Analysis	97
5.3.2	Karnaugh Maps	99
5.4	Combinational Building Blocks	102
5.4.1	Adders	102
5.4.2	Multiplexers	104
5.5	Counters and Registers	107
5.5.1	Counters	107
5.5.2	Shift Registers	110
5.6	Other Digital Logic Topics	112
5.6.1	Latches and Flip-Flops	112
5.6.2	Finite State Machines	113
5.6.3	Verilog	113
5.6.4	Programmable Logic Devices to Field-Programmable Gate Arrays	113
5.6.5	Practical Considerations	114
	References	114
CHAPTER 6	Networks and MPI for Cluster Computing	117
6.1	Why Message Passing/MPI?	118
6.1.1	Shared Memory Cache Coherent Nonuniform Memory Access Architecture does not Extend to an Extreme Scale	120
6.1.2	Message Passing Provides Scalability for "Grand Challenge" Problems	122
6.1.3	SPMD Model Enables Reasoning About and Programming $O(\text{Million})$ Process Programs	123
6.1.4	HPC in the Cloud Compared with Traditional HPC	125
6.2	The Message Passing Concept	127
6.2.1	Point-to-Point Communication	127
6.2.2	Introduction to Collective Communication	132
6.2.3	BSP Model Illustrated with a Simple Example	136
6.2.4	Nonblocking Communication	138

6.3	High-Performance Networks	140
6.3.1	Differences from Commodity Networks	142
6.3.2	Introduction to RDMA	144
6.3.3	Main Memory and RDMA	146
6.3.4	Ethernet RDMA	148
6.4	Advanced Concepts	148
6.4.1	Hybrid Programming with MPI and OpenMP	148
6.4.2	Supercomputers: Enabling HPC	150
	References	152

2 FOR STUDENTS

CHAPTER 7 Fork-Join Parallelism with a Data-Structures

Focus	157
7.1 Meta-Introduction: An Instructor's View of This Material	158
7.1.1 Where This Material Fits in a Changing Curriculum	158
7.1.2 Six Theses On A Successful Approach to this Material...	159
7.1.3 How to Use These Materials—And Improve Them	159
7.2 Introduction	160
7.2.1 More Than One Thing At Once	160
7.2.2 Parallelism Versus Concurrency Control	161
7.2.3 Basic Threads and Shared Memory	164
7.2.4 Other Models	170
7.3 Basic Fork-Join Parallelism	173
7.3.1 A Simple Example: Okay Idea, Inferior Style	173
7.3.2 Why Not Use One Thread Per Processor?	178
7.3.3 Divide-and-Conquer Parallelism	181
7.3.4 The Java ForkJoin Framework	187
7.3.5 Reductions and Maps	190
7.3.6 Data Structures Besides Arrays	193
7.4 Analyzing Fork-Join Algorithms	194
7.4.1 Work and Span	194
7.4.2 Amdahl's Law	199
7.4.3 Comparing Amdahl's Law and Moore's Law	201
7.5 Fancier Fork-Join Algorithms: Prefix, Pack, Sort	201
7.5.1 Parallel Prefix Sum	202
7.5.2 Pack	206
7.5.3 Parallel Quicksort	207
7.5.4 Parallel Mergesort	209
Acknowledgments	211
Reference	211

CHAPTER 8 Shared-Memory Concurrency Control with a Data-Structures Focus

8.1	Introduction	214
8.2	The Programming Model	214
8.3	Synchronization with Locks	216
8.3.1	The Need for Synchronization	216
8.3.2	Locks	220
8.3.3	Locks in Java	222
8.4	Race Conditions: Bad Interleavings and Data Races	226
8.4.1	Bad Interleavings: An Example with Stacks	227
8.4.2	Data Races: Wrong Even When They Look Right	232
8.5	Concurrency Programming Guidelines	237
8.5.1	Conceptually Splitting Memory into Three Parts	237
8.5.2	Approaches to Synchronization	240
8.6	Deadlock	245
8.7	Additional Synchronization Primitives	248
8.7.1	Reader/Writer Locks	249
8.7.2	Condition Variables	251
8.7.3	Other Primitives	258
	Acknowledgments	259
	Reference	259

CHAPTER 9 Parallel Computing in a Python-Based Computer Science Course

9.1	Parallel Programming	262
9.1.1	Parallelizable Loops	262
9.1.2	Barriers	263
9.1.3	Outline	264
9.2	Parallel Reduction	264
9.2.1	Reducing in Parallel when $n \leq p$	265
9.2.2	Reducing in Parallel when $n > p$	267
9.3	Parallel Scanning	269
9.3.1	Scanning in Parallel when $n \leq p$	270
9.3.2	Scanning in Parallel when $n > p$	272
9.3.3	Inclusive Scans in Parallel	274
9.4	Copy-Scans	274
9.4.1	Copy-Scanning in Parallel when $n \leq p$	275
9.4.2	Copy-Scanning in Parallel when $n > p$	275
9.5	Partitioning in Parallel	276
9.5.1	Meld Operations	277

9.5.2	Permute Operations	278
9.5.3	Partitioning	279
9.5.4	Analysis	281
9.6	Parallel Quicksort	281
9.6.1	Analysis	284
9.7	How to Perform Segmented Scans and Reductions	285
9.7.1	Segmented Scans	285
9.7.2	Segmented Inclusive Scans	289
9.7.3	Segmented Copy-Scans	289
9.7.4	Segmented Reductions	290
9.8	Comparing Sequential and Parallel Running Times	293
	References	297
 CHAPTER 10 Parallel Programming Illustrated Through Conway's		
	Game of Life	299
10.1	Introduction	300
10.1.1	Conway's Game of Life	300
10.1.2	Programming the Game of Life	301
10.1.3	General Thoughts on Parallelism	303
10.2	Parallel Variants	303
10.2.1	Data Parallelism	303
10.2.2	Loop-Based Parallelism	308
10.2.3	Coarse-Grained Data Parallelism	309
10.3	Advanced Topics	316
10.3.1	Data Partitioning	316
10.3.2	Combining Work, Minimizing Communication	318
10.3.3	Load Balancing	320
10.4	Summary	320
	References	321
Appendix A: Chapters and Topics		323
Index		329