Contents

Preface xv
Acknowledgments xix
Suggested ways to teach the art of multiprocessor programming xxi

CHAPTER 1 Introduction **1**
1.1 Shared objects and synchronization 3
1.2 A fable 6
1.2.1 Properties of a mutual exclusion protocol 8
1.2.2 The moral 9
1.3 The producer–consumer problem 9
1.4 The readers–writers problem 11
1.5 The harsh realities of parallelization 12
1.6 Parallel programming 14
1.7 Chapter notes 15
1.8 Exercises 15

PART 1 Principles

CHAPTER 2 Mutual exclusion **21**
2.1 Time and events 21
2.2 Critical sections 22
2.3 Two-thread solutions 25
2.3.1 The `LockOne` class 25
2.3.2 The `LockTwo` class 26
2.3.3 The Peterson lock 27
2.4 Notes on deadlock 29
2.5 The filter lock 30
2.6 Fairness 33
2.7 Lamport's Bakery algorithm 34
2.8 Bounded timestamps 35
2.9 Lower bounds on the number of locations 39
2.10 Chapter notes 41
2.11 Exercises 42

CHAPTER 3 Concurrent objects **49**
3.1 Concurrency and correctness 49
3.2 Sequential objects 52
3.3 Sequential consistency 53
3.3.1 Sequential consistency versus real-time order 55
3.3.2 Sequential consistency is nonblocking 56

3.3.3 Compositionality 57
3.4 Linearizability 58
3.4.1 Linearization points 58
3.4.2 Linearizability versus sequential consistency 59
3.5 Quiescent consistency 59
3.5.1 Properties of quiescent consistency 60
3.6 Formal definitions 60
3.6.1 Histories 60
3.6.2 Linearizability 61
3.6.3 Linearizability is compositional 63
3.6.4 Linearizability is nonblocking 63
3.7 Memory consistency models 64
3.8 Progress conditions 64
3.8.1 Wait-freedom 65
3.8.2 Lock-freedom 65
3.8.3 Obstruction-freedom 66
3.8.4 Blocking progress conditions 67
3.8.5 Characterizing progress conditions 67
3.9 Remarks 68
3.10 Chapter notes 69
3.11 Exercises 70

CHAPTER 4 Foundations of shared memory **75**
4.1 The space of registers 76
4.2 Register constructions 81
4.2.1 Safe MRSW registers 82
4.2.2 A regular Boolean MRSW register 83
4.2.3 A regular M-valued MRSW register 84
4.2.4 An atomic SRSW register 85
4.2.5 An atomic MRSW register 87
4.2.6 An atomic MRMW register 90
4.3 Atomic snapshots 92
4.3.1 An obstruction-free snapshot 92
4.3.2 A wait-free snapshot 93
4.3.3 Correctness arguments 97
4.4 Chapter notes 98
4.5 Exercises 99

CHAPTER 5 The relative power of primitive synchronization operations **103**
5.1 Consensus numbers 104
5.1.1 States and valence 105
5.2 Atomic registers 106
5.3 Consensus protocols 109
5.4 FIFO queues 110

5.5 Multiple assignment objects 113
5.6 Read–modify–write operations 116
5.7 Common2 RMW operations 117
5.8 The `compareAndSet` operation 119
5.9 Chapter notes 120
5.10 Exercises 121

CHAPTER 6 Universality of consensus 129
6.1 Introduction 129
6.2 Universality 130
6.3 A lock-free universal construction 130
6.4 A wait-free universal construction 134
6.5 Chapter notes 140
6.6 Exercises 141

PART 2 Practice

CHAPTER 7 Spin locks and contention 147
7.1 Welcome to the real world 147
7.2 Volatile fields and atomic objects 150
7.3 Test-and-set locks 150
7.4 Exponential back-off 154
7.5 Queue locks 156
7.5.1 Array-based locks 156
7.5.2 The CLH queue lock 159
7.5.3 The MCS queue lock 161
7.6 A queue lock with timeouts 163
7.7 Hierarchical locks 166
7.7.1 A hierarchical back-off lock 167
7.7.2 Cohort locks 167
7.7.3 A cohort lock implementation 170
7.8 A composite lock 171
7.9 A fast path for threads running alone 178
7.10 One lock to rule them all 180
7.11 Chapter notes 180
7.12 Exercises 181

CHAPTER 8 Monitors and blocking synchronization 183
8.1 Introduction 183
8.2 Monitor locks and conditions 183
8.2.1 Conditions 185
8.2.2 The lost-wakeup problem 187
8.3 Readers–writers locks 189
8.3.1 Simple readers–writers lock 190
8.3.2 Fair readers–writers lock 192

8.4 Our own reentrant lock . . . 194
8.5 Semaphores . . . 194
8.6 Chapter notes . . . 196
8.7 Exercises . . . 197

CHAPTER 9 Linked lists: The role of locking . . . 201
9.1 Introduction . . . 201
9.2 List-based sets . . . 202
9.3 Concurrent reasoning . . . 204
9.4 Coarse-grained synchronization . . . 206
9.5 Fine-grained synchronization . . . 207
9.6 Optimistic synchronization . . . 211
9.7 Lazy synchronization . . . 215
9.8 Nonblocking synchronization . . . 220
9.9 Discussion . . . 225
9.10 Chapter notes . . . 226
9.11 Exercises . . . 226

CHAPTER 10 Queues, memory management, and the ABA problem . . . 229
10.1 Introduction . . . 229
10.2 Queues . . . 230
10.3 A bounded partial queue . . . 230
10.4 An unbounded total queue . . . 235
10.5 A lock-free unbounded queue . . . 236
10.6 Memory reclamation and the ABA problem . . . 240
10.6.1 A naïve synchronous queue . . . 244
10.7 Dual data structures . . . 244
10.8 Chapter notes . . . 248
10.9 Exercises . . . 248

CHAPTER 11 Stacks and elimination . . . 251
11.1 Introduction . . . 251
11.2 An unbounded lock-free stack . . . 251
11.3 Elimination . . . 254
11.4 The elimination back-off stack . . . 255
11.4.1 A lock-free exchanger . . . 255
11.4.2 The elimination array . . . 257
11.5 Chapter notes . . . 260
11.6 Exercises . . . 261

CHAPTER 12 Counting, sorting, and distributed coordination . . . 265
12.1 Introduction . . . 265
12.2 Shared counting . . . 265
12.3 Software combining . . . 266
12.3.1 Overview . . . 267

12.3.2 An extended example 274
12.3.3 Performance and robustness 275
12.4 Quiescently consistent pools and counters 276
12.5 Counting networks 276
12.5.1 Networks that count 276
12.5.2 The bitonic counting network 279
12.5.3 Performance and pipelining 287
12.6 Diffracting trees 288
12.7 Parallel sorting 292
12.8 Sorting networks 293
12.8.1 Designing a sorting network 294
12.9 Sample sorting 296
12.10 Distributed coordination 298
12.11 Chapter notes 299
12.12 Exercises 300

CHAPTER 13 Concurrent hashing and natural parallelism 305
13.1 Introduction 305
13.2 Closed-address hash sets 306
13.2.1 A coarse-grained hash set 308
13.2.2 A striped hash set 310
13.2.3 A refinable hash set 311
13.3 A lock-free hash set 315
13.3.1 Recursive split-ordering 315
13.3.2 The `BucketList` class 318
13.3.3 The `LockFreeHashSet<T>` class 319
13.4 An open-address hash set 323
13.4.1 Cuckoo hashing 323
13.4.2 Concurrent cuckoo hashing 324
13.4.3 Striped concurrent cuckoo hashing 329
13.4.4 A refinable concurrent cuckoo hash set 331
13.5 Chapter notes 332
13.6 Exercises 334

CHAPTER 14 Skiplists and balanced search 335
14.1 Introduction 335
14.2 Sequential skiplists 335
14.3 A lock-based concurrent skiplist 337
14.3.1 A bird's-eye view 337
14.3.2 The algorithm 339
14.4 A lock-free concurrent skiplist 345
14.4.1 A bird's-eye view 345
14.4.2 The algorithm in detail 348
14.5 Concurrent skiplists 355
14.6 Chapter notes 356

14.7 Exercises 356

CHAPTER 15 Priority queues 359
15.1 Introduction 359
15.1.1 Concurrent priority queues 359
15.2 An array-based bounded priority queue 360
15.3 A tree-based bounded priority queue 361
15.4 An unbounded heap-based priority queue 363
15.4.1 A sequential heap 363
15.4.2 A concurrent heap 365
15.5 A skiplist-based unbounded priority queue 371
15.6 Chapter notes 374
15.7 Exercises 375

CHAPTER 16 Scheduling and work distribution 377
16.1 Introduction 377
16.2 Analyzing parallelism 384
16.3 Realistic multiprocessor scheduling 387
16.4 Work distribution 389
16.4.1 Work stealing 389
16.4.2 Yielding and multiprogramming 390
16.5 Work-stealing deques 390
16.5.1 A bounded work-stealing deque 391
16.5.2 An unbounded work-stealing deque 395
16.5.3 Work dealing 397
16.6 Chapter notes 400
16.7 Exercises 401

CHAPTER 17 Data parallelism 405
17.1 MapReduce 407
17.1.1 The `MapReduce` framework 408
17.1.2 A `MapReduce`-based `WordCount` application 410
17.1.3 A `MapReduce`-based `KMeans` application 411
17.1.4 The `MapReduce` implementation 411
17.2 Stream computing 414
17.2.1 A stream-based `WordCount` application 416
17.2.2 A stream-based `KMeans` application 417
17.2.3 Making aggregate operations parallel 419
17.3 Chapter notes 422
17.4 Exercises 423

CHAPTER 18 Barriers 431
18.1 Introduction 431
18.2 Barrier implementations 432
18.3 Sense reversing barrier 433
18.4 Combining tree barrier 434

18.5 Static tree barrier 436
18.6 Termination detection barriers 438
18.7 Chapter notes 442
18.8 Exercises 443

CHAPTER 19 Optimism and manual memory management **451**
19.1 Transitioning from Java to C++ 451
19.2 Optimism and explicit reclamation 451
19.3 Protecting pending operations 454
19.4 An object for managing memory 455
19.5 Traversing a list 456
19.6 Hazard pointers 458
19.7 Epoch-based reclamation 462
19.8 Chapter notes 465
19.9 Exercises 466

CHAPTER 20 Transactional programming **467**
20.1 Challenges in concurrent programming 467
20.1.1 Problems with locking 467
20.1.2 Problems with explicit speculation 468
20.1.3 Problems with nonblocking algorithms 470
20.1.4 Problems with compositionality 471
20.1.5 Summary 472
20.2 Transactional programming 472
20.2.1 An example of transactional programming 473
20.3 Hardware support for transactional programming 475
20.3.1 Hardware speculation 475
20.3.2 Basic cache coherence 475
20.3.3 Transactional cache coherence 476
20.3.4 Limitations of hardware support 477
20.4 Transactional lock elision 478
20.4.1 Discussion 480
20.5 Transactional memory 481
20.5.1 Run-time scheduling 482
20.5.2 Explicit self-abort 483
20.6 Software transactions 483
20.6.1 Transactions with ownership records 485
20.6.2 Transactions with value-based validation 490
20.7 Combining hardware and software transactions 492
20.8 Transactional data structure design 493
20.9 Chapter notes 494
20.10 Exercises 494

APPENDIX A Software basics **497**
A.1 Introduction 497
A.2 Java 497

A.2.1 Threads 497
A.2.2 Monitors 498
A.2.3 Yielding and sleeping 501
A.2.4 Thread-local objects 502
A.2.5 Randomization 503
A.3 The Java memory model 504
A.3.1 Locks and synchronized blocks 505
A.3.2 Volatile fields 506
A.3.3 Final fields 506
A.4 C++ 508
A.4.1 Threads in C++ 508
A.4.2 Locks in C++ 509
A.4.3 Condition variables 510
A.4.4 Atomic variables 512
A.4.5 Thread-local storage 513
A.5 C# 514
A.5.1 Threads 514
A.5.2 Monitors 515
A.5.3 Thread-local objects 517
A.6 Appendix notes 518

APPENDIX B Hardware basics 519
B.1 Introduction (and a puzzle) 519
B.2 Processors and threads 522
B.3 Interconnect 522
B.4 Memory 523
B.5 Caches 523
B.5.1 Coherence 524
B.5.2 Spinning 526
B.6 Cache-conscious programming, or the puzzle solved 526
B.7 Multicore and multithreaded architectures 527
B.7.1 Relaxed memory consistency 528
B.8 Hardware synchronization instructions 529
B.9 Appendix notes 530
B.10 Exercises 531

Bibliography 533
Index 541