

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/213876653>


The Art of Multiprocessor Programming

Book · January 2008
DOI: 10.1145/1146381.1146382 · Source: DBLP


CITATIONS
779

READS
4,516

2 authors:




[Maurice Herlihy](#)
Brown University
333 PUBLICATIONS **17,436** CITATIONS
[SEE PROFILE](#)



[Nir Shavit](#)
Massachusetts Institute of Technology
191 PUBLICATIONS **8,063** CITATIONS
[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Distributed Transactional Memory [View project](#)

The Art of Multiprocessor Programming

Maurice Herlihy

Nir Shavit



AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Morgan Kaufmann Publishers is an imprint of Elsevier




MORGAN KAUFMANN PUBLISHERS

Contents

Acknowledgments	xvii
Preface	xix
I Introduction	I
1.1 Shared Objects and Synchronization	3
1.2 A Fable	6
1.2.1 Properties of Mutual Exclusion	8
1.2.2 The Moral	9
1.3 The Producer–Consumer Problem	10
1.4 The Readers–Writers Problem	12
1.5 The Harsh Realities of Parallelization	13
1.6 Parallel Programming	15
1.7 Chapter Notes	15
1.8 Exercises	16
PRINCIPLES	19
2 Mutual Exclusion	21
2.1 Time	21
2.2 Critical Sections	22
	vii

2.3	2-Thread Solutions	24
2.3.1	The LockOne Class	25
2.3.2	The LockTwo Class	26
2.3.3	The Peterson Lock	27
2.4	The Filter Lock	28
2.5	Fairness	31
2.6	Lamport's Bakery Algorithm	31
2.7	Bounded Timestamps	33
2.8	Lower Bounds on the Number of Locations	37
2.9	Chapter Notes	40
2.10	Exercises	41
3	Concurrent Objects	45
3.1	Concurrency and Correctness	45
3.2	Sequential Objects	48
3.3	Quiescent Consistency	49
3.3.1	Remarks	51
3.4	Sequential Consistency	51
3.4.1	Remarks	52
3.5	Linearizability	54
3.5.1	Linearization Points	55
3.5.2	Remarks	55
3.6	Formal Definitions	55
3.6.1	Linearizability	57
3.6.2	Compositional Linearizability	57
3.6.3	The Nonblocking Property	58
3.7	Progress Conditions	59
3.7.1	Dependent Progress Conditions	60
3.8	The Java Memory Model	61
3.8.1	Locks and Synchronized Blocks	62
3.8.2	Volatile Fields	63
3.8.3	Final Fields	63

3.9 Remarks	64
3.10 Chapter Notes	65
3.11 Exercises	66
4 Foundations of Shared Memory	71
4.1 The Space of Registers	72
4.2 Register Constructions	77
4.2.1 MRSW Safe Registers	78
4.2.2 A Regular Boolean MRSW Register	78
4.2.3 A Regular M -Valued MRSW Register	79
4.2.4 An Atomic SRSW Register	81
4.2.5 An Atomic MRSW Register	82
4.2.6 An Atomic MRMW Register	85
4.3 Atomic Snapshots	87
4.3.1 An Obstruction-Free Snapshot	87
4.3.2 A Wait-Free Snapshot	88
4.3.3 Correctness Arguments	90
4.4 Chapter Notes	93
4.5 Exercises	94
5 The Relative Power of Primitive Synchronization Operations	99
5.1 Consensus Numbers	100
5.1.1 States and Valence	101
5.2 Atomic Registers	103
5.3 Consensus Protocols	106
5.4 FIFO Queues	106
5.5 Multiple Assignment Objects	110
5.6 Read-Modify-Write Operations	112
5.7 Common2 RMW Operations	114
5.8 The compareAndSet () Operation	116
5.9 Chapter Notes	117
5.10 Exercises	118

6	Universality of Consensus	125
6.1	Introduction	125
6.2	Universality	126
6.3	A Lock-Free Universal Construction	126
6.4	A Wait-Free Universal Construction	130
6.5	Chapter Notes	136
6.6	Exercises	137
	PRACTICE	139
7	Spin Locks and Contention	141
7.1	Welcome to the Real World	141
7.2	Test-And-Set Locks	144
7.3	TAS-Based Spin Locks Revisited	146
7.4	Exponential Backoff	147
7.5	Queue Locks	149
7.5.1	Array-Based Locks	150
7.5.2	The CLH Queue Lock	151
7.5.3	The MCS Queue Lock	154
7.6	A Queue Lock with Timeouts	157
7.7	A Composite Lock	159
7.7.1	A Fast-Path Composite Lock	165
7.8	Hierarchical Locks	167
7.8.1	A Hierarchical Backoff Lock	167
7.8.2	A Hierarchical CLH Queue Lock	168
7.9	One Lock To Rule Them All	173
7.10	Chapter Notes	173
7.11	Exercises	174
8	Monitors and Blocking Synchronization	177
8.1	Introduction	177

8.2	Monitor Locks and Conditions	178
8.2.1	Conditions	179
8.2.2	The Lost-Wakeup Problem	181
8.3	Readers–Writers Locks	183
8.3.1	Simple Readers–Writers Lock	184
8.3.2	Fair Readers–Writers Lock	185
8.4	Our Own Reentrant Lock	187
8.5	Semaphores	189
8.6	Chapter Notes	189
8.7	Exercises	190
9	Linked Lists: The Role of Locking	195
9.1	Introduction	195
9.2	List-Based Sets	196
9.3	Concurrent Reasoning	198
9.4	Coarse-Grained Synchronization	200
9.5	Fine-Grained Synchronization	201
9.6	Optimistic Synchronization	205
9.7	Lazy Synchronization	208
9.8	Non-Blocking Synchronization	213
9.9	Discussion	218
9.10	Chapter Notes	219
9.11	Exercises	219
10	Concurrent Queues and the ABA Problem	223
10.1	Introduction	223
10.2	Queues	224
10.3	A Bounded Partial Queue	225
10.4	An Unbounded Total Queue	229
10.5	An Unbounded Lock-Free Queue	230
10.6	Memory Reclamation and the ABA Problem	233
10.6.1	A Naïve Synchronous Queue	237

10.7 Dual Data Structures	238
10.8 Chapter Notes	241
10.9 Exercises	241
11 Concurrent Stacks and Elimination	245
11.1 Introduction	245
11.2 An Unbounded Lock-Free Stack	245
11.3 Elimination	248
11.4 The Elimination Backoff Stack	249
11.4.1 A Lock-Free Exchanger	249
11.4.2 The Elimination Array	251
11.5 Chapter Notes	255
11.6 Exercises	255
12 Counting, Sorting, and Distributed Coordination	259
12.1 Introduction	259
12.2 Shared Counting	259
12.3 Software Combining	260
12.3.1 Overview	261
12.3.2 An Extended Example	267
12.3.3 Performance and Robustness	269
12.4 Quiescently Consistent Pools and Counters	269
12.5 Counting Networks	270
12.5.1 Networks That Count	270
12.5.2 The Bitonic Counting Network	273
12.5.3 Performance and Pipelining	280
12.6 Diffracting Trees	282
12.7 Parallel Sorting	286
12.8 Sorting Networks	286
12.8.1 Designing a Sorting Network	287
12.9 Sample Sorting	290
12.10 Distributed Coordination	291

12.11 Chapter Notes	292
12.12 Exercises	293
13 Concurrent Hashing and Natural Parallelism	299
13.1 Introduction	299
13.2 Closed-Address Hash Sets	300
13.2.1 A Coarse-Grained Hash Set	302
13.2.2 A Striped Hash Set	303
13.2.3 A Refinable Hash Set	305
13.3 A Lock-Free Hash Set	309
13.3.1 Recursive Split-Ordering	309
13.3.2 The <code>BucketList</code> Class	312
13.3.3 The <code>LockFreeHashSet<T></code> Class	313
13.4 An Open-Addressed Hash Set	316
13.4.1 Cuckoo Hashing	316
13.4.2 Concurrent Cuckoo Hashing	318
13.4.3 Striped Concurrent Cuckoo Hashing	322
13.4.4 A Refinable Concurrent Cuckoo Hash Set	324
13.5 Chapter Notes	325
13.6 Exercises	326
14 Skiplists and Balanced Search	329
14.1 Introduction	329
14.2 Sequential Skiplists	329
14.3 A Lock-Based Concurrent Skiplist	331
14.3.1 A Bird's-Eye View	331
14.3.2 The Algorithm	333
14.4 A Lock-Free Concurrent Skiplist	339
14.4.1 A Bird's-Eye View	339
14.4.2 The Algorithm in Detail	341
14.5 Concurrent Skiplists	348
14.6 Chapter Notes	348
14.7 Exercises	349

15 Priority Queues	351
15.1 Introduction	351
15.1.1 Concurrent Priority Queues	351
15.2 An Array-Based Bounded Priority Queue	352
15.3 A Tree-Based Bounded Priority Queue	353
15.4 An Unbounded Heap-Based Priority Queue	355
15.4.1 A Sequential Heap	356
15.4.2 A Concurrent Heap	357
15.5 A Skiplist-Based Unbounded Priority Queue	363
15.6 Chapter Notes	366
15.7 Exercises	366
16 Futures, Scheduling, and Work Distribution	369
16.1 Introduction	369
16.2 Analyzing Parallelism	375
16.3 Realistic Multiprocessor Scheduling	378
16.4 Work Distribution	381
16.4.1 Work Stealing	381
16.4.2 Yielding and Multiprogramming	381
16.5 Work-Stealing Dequeues	382
16.5.1 A Bounded Work-Stealing Dequeue	383
16.5.2 An Unbounded Work-Stealing DEQueue	386
16.5.3 Work Balancing	390
16.6 Chapter Notes	392
16.7 Exercises	392
17 Barriers	397
17.1 Introduction	397
17.2 Barrier Implementations	398
17.3 Sense-Reversing Barrier	399
17.4 Combining Tree Barrier	401
17.5 Static Tree Barrier	402
17.6 Termination Detecting Barriers	404

17.7 Chapter Notes	408
17.8 Exercises	409
18 Transactional Memory	417
18.1 Introduction	417
18.1.1 What is Wrong with Locking?	417
18.1.2 What is Wrong with <code>compareAndSet()</code> ?	418
18.1.3 What is Wrong with Compositionality?	420
18.1.4 What can We Do about It?	421
18.2 Transactions and Atomicity	421
18.3 Software Transactional Memory	424
18.3.1 Transactions and Transactional Threads	427
18.3.2 Zombies and Consistency	428
18.3.3 Atomic Objects	429
18.3.4 Dependent or Independent Progress?	431
18.3.5 Contention Managers	431
18.3.6 Implementing Atomic Objects	433
18.3.7 An Obstruction-Free Atomic Object	434
18.3.8 A Lock-Based Atomic Object	438
18.4 Hardware Transactional Memory	445
18.4.1 Cache Coherence	446
18.4.2 Transactional Cache Coherence	447
18.4.3 Enhancements	447
18.5 Chapter Notes	448
18.6 Exercises	449



APPENDIX

451

A Software Basics	453
A.1 Introduction	453
A.2 Java	453
A.2.1 Threads	453
A.2.2 Monitors	455
A.2.3 Yielding and Sleeping	458
A.2.4 Thread-Local Objects	458

A.3 C#	460
A.3.1 Threads	460
A.3.2 Monitors	461
A.3.3 Thread-Local Objects	462
A.4 Pthreads	464
A.4.1 Thread-Local Storage	465
A.5 Chapter Notes	466
B Hardware Basics	469
B.1 Introduction (and a Puzzle)	469
B.2 Processors and Threads	472
B.3 Interconnect	472
B.4 Memory	473
B.5 Caches	473
B.5.1 Coherence	474
B.5.2 Spinning	476
B.6 Cache-Conscious Programming, or the Puzzle Solved	476
B.7 Multi-Core and Multi-Threaded Architectures	477
B.7.1 Relaxed Memory Consistency	478
B.8 Hardware Synchronization Instructions	479
B.9 Chapter Notes	481
B.10 Exercises	481
Bibliography	483
Index	495