

Contents

Introduction	xxi
Chapter 1: The New Architecture	1
What Is a Multicore?	2
Multicore Architectures	2
Hybrid Multicore Architectures	3
The Software Developer's Viewpoint	4
The Basic Processor Architecture	5
The CPU (Instruction Set)	7
Memory Is the Key	9
Registers	11
Cache	12
Main Memory	13
The Bus Connection	14
From Single Core to Multicore	15
Multiprogramming and Multiprocessing	15
Parallel Programming	15
Multicore Application Design and Implementation	16
Summary	17
Chapter 2: Four Effective Multicore Designs	19
The AMD Multicore Opteron	21
Opteron's Direct Connect and HyperTransport	22
System Request Interface and Crossbar	23
The Opteron Is NUMA	24
Cache and the Multiprocessor Opteron	25
The Sun UltraSparc T1 Multiprocessor	25
Program Profile 2-1	26
UltraSparc T1 Cores	27
Cross Talk and The Crossbar	27
DDRAM Controller and L2 Cache	28
UltraSparc T1 and the Sun and GNU gcc Compilers	28

Contents

The IBM Cell Broadband Engine	28
CBE and Linux	29
CBE Memory Models	29
Hidden from the Operating System	30
Synergistic Processor Unit	31
Intel Core 2 Duo Processor	31
Northbridge and Southbridge	32
Intel's PCI Express	32
Core 2 Duo's Instruction Set	32
Summary	33
Chapter 3: The Challenges of Multicore Programming	35
What Is the Sequential Model?	36
What Is Concurrency?	37
Software Development	37
Challenge #1: Software Decomposition	41
Challenge #2: Task-to-Task Communication	47
Challenge #3: Concurrent Access to Data or Resources by Multiple Tasks or Agents	51
Challenge #4: Identifying the Relationships between Concurrently Executing Tasks	56
Challenge #5: Controlling Resource Contention Between Tasks	59
Challenge #6: How Many Processes or Threads Are Enough?	59
Challenges #7 and #8: Finding Reliable and Reproducible Debugging and Testing	60
Challenge #9: Communicating a Design That Has Multiprocessing Components	61
Challenge #10: Implementing Multiprocessing and Multithreading in C++	62
C++ Developers Have to Learn New Libraries	63
Processor Architecture Challenges	64
Summary	64
Chapter 4: The Operating System's Role	67
What Part Does the Operating System Play?	68
Providing a Consistent Interface	68
Managing Hardware Resources and Other Software Applications	68
The Developer's Interaction with the Operating System	69
Core Operating System Services	71
The Application Programmer's Interface	75
Decomposition and the Operating System's Role	83
Hiding the Operating System's Role	86
Taking Advantage of C++ Power of Abstraction and Encapsulation	86
Interface Classes for the POSIX APIs	86
Summary	93

Chapter 5: Processes, C++ Interface Classes, and Predicates	95
We Say <i>Multicore</i>, We Mean <i>Multiprocessor</i>	96
What Is a Process?	97
Why Processes and Not Threads?	97
Using <code>posix_spawn()</code>	98
The <code>file_actions</code> Parameter	99
The <code>attrp</code> Parameter	100
A Simple <code>posix_spawn()</code> Example	103
The <code>guess_it</code> Program Using <code>posix_spawn</code>	104
Who Is the Parent? Who Is the Child?	107
Processes: A Closer Look	108
Process Control Block	108
Anatomy of a Process	109
Process States	111
How Are Processes Scheduled?	114
Monitoring Processes with the <code>ps</code> Utility	116
Setting and Getting Process Priorities	119
What Is a Context Switch?	121
The Activities in Process Creation	121
Using the <code>fork()</code> Function Call	122
Using the <code>exec()</code> Family of System Calls	122
Working with Process Environment Variables	126
Using <code>system()</code> to Spawn Processes	127
Killing a Process	127
The <code>exit()</code> , and <code>abort()</code> Calls	128
The <code>kill()</code> Function	128
Process Resources	129
Types of Resources	130
POSIX Functions to Set Resource Limits	131
What Are Asynchronous and Synchronous Processes	133
Synchronous vs. Asynchronous Processes for <code>fork()</code> , <code>posix_spawn()</code> , <code>system()</code> , and <code>exec()</code>	135
The <code>wait()</code> Function Call	135
Predicates, Processes, and Interface Classes	137
Program Profile 5-1	138
Summary	141
Chapter 6: Multithreading	143
What Is a Thread?	143
User- and Kernel-Level Threads	144
Thread Context	147

Contents

Hardware Threads and Software Threads	149
Thread Resources	149
Comparing Threads to Processes	150
Context Switching	150
Throughput	150
Communicating between Entities	150
Corrupting Process Data	151
Killing the Entire Process	151
Reuse by Other Programs	151
Key Similarities and Differences between Threads and Processes	152
Setting Thread Attributes	153
The Architecture of a Thread	155
Thread States	156
Scheduling and Thread Contention Scope	157
Scheduling Policy and Priority	159
Scheduling Allocation Domains	160
A Simple Threaded Program	160
Compiling and Linking Threaded Programs	162
Creating Threads	162
Passing Arguments to a Thread	163
Program Profile 6-1	165
Joining Threads	165
Getting the Thread Id	166
Using the Pthread Attribute Object	167
Managing Threads	171
Terminating Threads	171
Managing the Thread's Stack	180
Setting Thread Scheduling and Priorities	183
Setting Contention Scope of a Thread	187
Using sysconf()	188
Thread Safety and Libraries	190
Extending the Thread Interface Class	193
Program Profile 6-2	200
Summary	200
Chapter 7: Communication and Synchronization of Concurrent Tasks	203
Communication and Synchronization	204
Dependency Relationships	205
Counting Tasks Dependencies	208
What Is Interprocess Communication?	210
What Are Interthread Communications?	230

Synchronizing Concurrency	238
Types of Synchronization	239
Synchronizing Access to Data	240
Synchronization Mechanisms	246
Thread Strategy Approaches	268
Delegation Model	269
Peer-to-Peer Model	271
Producer-Consumer Model	272
Pipeline Model	273
SPMD and MPMD for Threads	274
Decomposition and Encapsulation of Work	276
Problem Statement	276
Strategy	276
Observation	277
Problem and Solution	277
Simple Agent Model Example of a Pipeline	278
Summary	282
Chapter 8: PADL and PBS: Approaches to Application Design	283
Designing Applications for Massive Multicore Processors	284
What Is PADL?	287
Layer 5: Application Architecture Selection	290
Layer 4: Concurrency Models in PADL	300
Layer 3: The Implementation Model of PADL	304
The Predicate Breakdown Structure (PBS)	326
An Example: PBS for the “Guess-My-Code” Game	327
Connecting PBS, PADL, and the SDLC	328
Coding the PBS	328
Summary	328
Chapter 9: Modeling Software Systems That Require Concurrency	331
What Is UML?	332
Modeling the Structure of a System	334
The Class Model	334
Visualizing Classes	336
Ordering the Attributes and Services	343
Visualizing Instances of a Class	345
Visualizing Template Classes	348
Showing the Relationship between Classes and Objects	349
Visualizing Interface Classes	353
The Organization of Interactive Objects	356

Contents

UML and Concurrent Behavior	357
Collaborating Objects	357
Multitasking and Multithreading with Processes and Threads	359
Message Sequences between Objects	361
The Activities of Objects	363
State Machines	365
Visualizing the Whole System	371
Summary	372
Chapter 10: Testing and Logical Fault Tolerance for Parallel Programs	375
Can You Just Skip the Testing?	376
Five Concurrency Challenges That Must Be Checked during Testing	377
Failure: The Result of Defects and Faults	379
Basic Testing Types	379
Defect Removal versus Defect Survival	380
How Do You Approach Defect Removal for Parallel Programs?	381
The Problem Statement	382
A Simple Strategy and Rough-Cut Solution Model	382
A Revised Solution Model Using Layer 5 from PADL	382
The PBS of the Agent Solution Model	383
What Are the Standard Software Engineering Tests?	386
Software Verification and Validation	387
The Code Doesn't Work — Now What?	388
What Is Logical Fault Tolerance?	391
Predicate Exceptions and Possible Worlds	397
What Is Model Checking?	398
Summary	398
Appendix A: UML for Concurrent Design	401
Appendix B: Concurrency Models	411
Appendix C: POSIX Standard for Thread Management	427
Appendix D: POSIX Standard for Process Management	567
Bibliography	593
Index	597