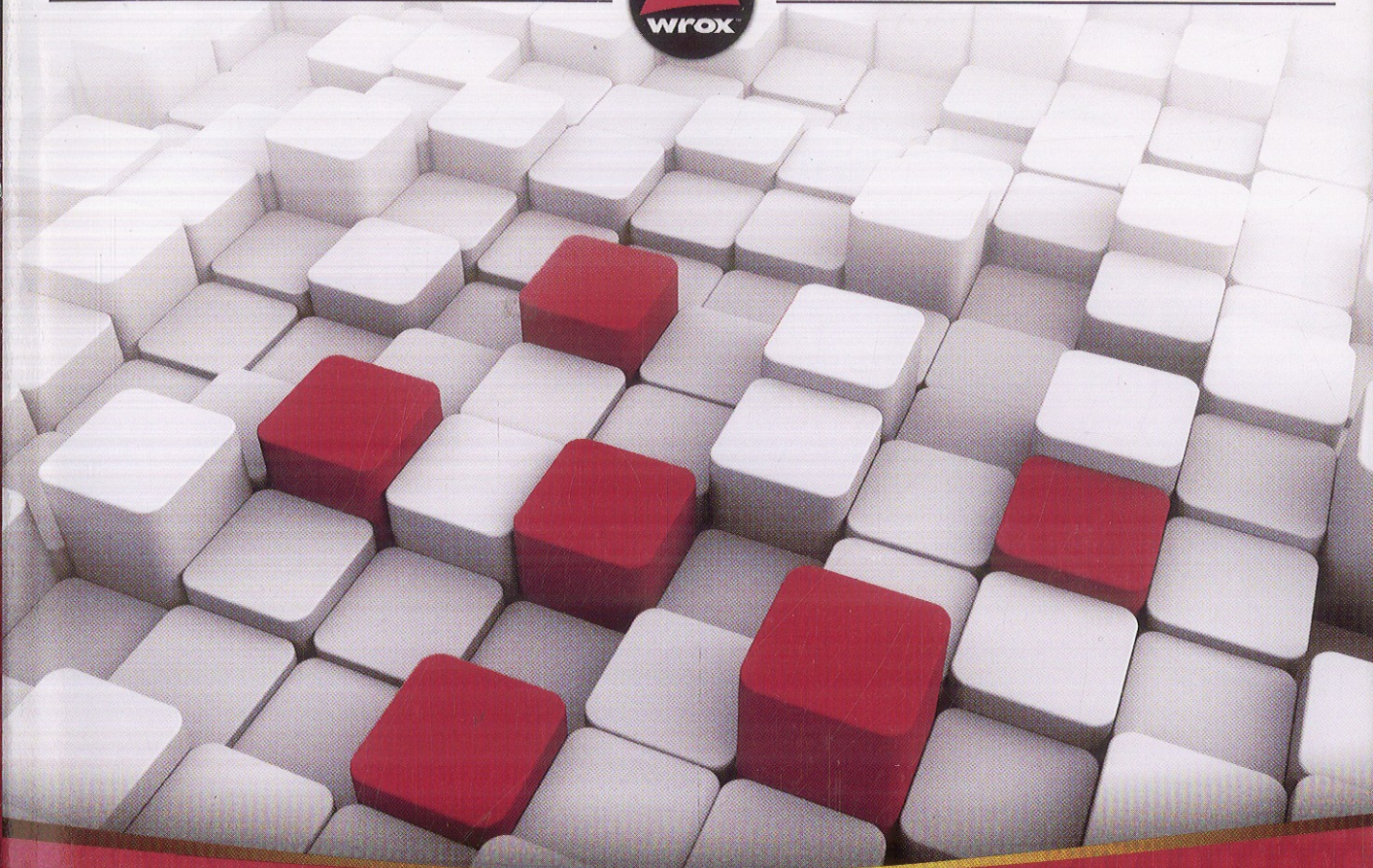


Join the discussion @ p2p.wrox.com



Wrox **Programmer to Programmer™**



Professional Parallel Programming with C#

Master Parallel Extensions with .NET 4

Gastón C. Hillar

CONTENTS

FOREWORD	<i>xix</i>
INTRODUCTION	<i>xxi</i>
CHAPTER 1: TASK-BASED PROGRAMMING	1
Working with Shared-Memory Multicore	2
Differences Between Shared-Memory Multicore and Distributed-Memory Systems	3
Parallel Programming and Multicore Programming	4
Understanding Hardware Threads and Software Threads	5
Understanding Amdahl's Law	10
Considering Gustafson's Law	13
Working with Lightweight Concurrency	16
Creating Successful Task-Based Designs	17
Designing With Concurrency in Mind	18
Understanding the Differences between Interleaved Concurrency, Concurrency, and Parallelism	19
Parallelizing Tasks	19
Minimizing Critical Sections	21
Understanding Rules for Parallel Programming for Multicore	22
Preparing for NUMA and Higher Scalability	22
Deciding the Convenience of Going Parallel	27
Summary	28
CHAPTER 2: IMPERATIVE DATA PARALLELISM	29
Launching Parallel Tasks	30
System.Threading.Tasks.Parallel Class	31
Parallel.Invoke	32
No Specific Execution Order	33
Advantages and Trade-Offs	37
Interleaved Concurrency and Concurrency	38
Transforming Sequential Code to Parallel Code	40
Detecting Parallelizable Hotspots	40
Measuring Speedups Achieved by Parallel Execution	43
Understanding the Concurrent Execution	45
Parallelizing Loops	45

Parallel.For	46
Refactoring an Existing Sequential Loop	48
Measuring Scalability	50
Working with Embarrassingly Parallel Problems	52
Parallel.ForEach	52
Working with Partitions in a Parallel Loop	54
Optimizing the Partitions According to the Number of Cores	56
Working with IEnumerable Sources of Data	58
Exiting from Parallel Loops	60
Understanding ParallelLoopState	62
Analyzing the Results of a Parallel Loop Execution	63
Catching Exceptions that Occur Inside Parallel Loops	64
Specifying the Desired Degree of Parallelism	66
ParallelOptions	66
Counting Hardware Threads	69
Logical Cores Aren't Physical Cores	70
Using Gantt Charts to Detect Critical Sections	71
Summary	72
CHAPTER 3: IMPERATIVE TASK PARALLELISM	73
<hr/>	
Creating and Managing Tasks	74
System.Threading.Tasks.Task	75
Understanding a Task's Status and Lifecycle	77
TaskStatus: Initial States	77
TaskStatus: Final States	78
Using Tasks to Parallelize Code	78
Starting Tasks	79
Visualizing Tasks Using Parallel Tasks and Parallel Stacks	80
Waiting for Tasks to Finish	85
Forgetting About Complex Threads	85
Cancelling Tasks Using Tokens	86
CancellationTokenSource	89
CancellationToken	89
TaskFactory	90
Handling Exceptions Thrown by Tasks	91
Returning Values from Tasks	92
TaskCreationOptions	95
Chaining Multiple Tasks Using Continuations	95
Mixing Parallel and Sequential Code with Continuations	97
Working with Complex Continuations ^{4.4}	97
TaskContinuationOptions	98

Programming Complex Parallel Algorithms with Critical Sections Using Tasks	100
Preparing the Code for Concurrency and Parallelism	101
Summary	101
CHAPTER 4: CONCURRENT COLLECTIONS	103
<hr/>	
Understanding the Features Offered by Concurrent Collections	104
<i>System.Collections.Concurrent</i>	107
ConcurrentQueue	107
Understanding a Parallel Producer-Consumer Pattern	111
Working with Multiple Producers and Consumers	115
Designing Pipelines by Using Concurrent Collections	120
ConcurrentStack	121
Transforming Arrays and Unsafe Collections into Concurrent Collections	128
ConcurrentBag	129
IProducerConsumerCollection	136
BlockingCollection	137
Cancelling Operations on a BlockingCollection	142
Implementing a Filtering Pipeline with Many BlockingCollection Instances	144
ConcurrentDictionary	150
Summary	155
CHAPTER 5: COORDINATION DATA STRUCTURES	157
<hr/>	
Using Cars and Lanes to Understand the Concurrency Nightmares	158
Undesired Side Effects	158
Race Conditions	159
Deadlocks	160
A Lock-Free Algorithm with Atomic Operations	161
A Lock-Free Algorithm with Local Storage	162
Understanding New Synchronization Mechanisms	163
Working with Synchronization Primitives	164
Synchronizing Concurrent Tasks with Barriers	165
Barrier and ContinueWhenAll	171
Catching Exceptions in all Participating Tasks	172
Working with Timeouts	173
Working with a Dynamic Number of Participants	178
Working with Mutual-Exclusion Locks	179
Working with Monitor	182

Working with Timeouts for Locks	184
Refactoring Code to Avoid Locks	187
Using Spin Locks as Mutual-Exclusion Lock Primitives	190
Working with Timeouts	193
Working with Spin-Based Waiting	194
Spinning and Yielding	197
Using the Volatile Modifier	200
Working with Lightweight Manual Reset Events	201
Working with ManualResetEventSlim to Spin and Wait	201
Working with Timeouts and Cancellations	206
Working with ManualResetEvent	210
Limiting Concurrency to Access a Resource	211
Working with SemaphoreSlim	212
Working with Timeouts and Cancellations	216
Working with Semaphore	216
Simplifying Dynamic Fork and Join Scenarios with CountdownEvent	219
Working with Atomic Operations	223
Summary	228

CHAPTER 6: PLINQ: DECLARATIVE DATA PARALLELISM **229**

Transforming LINQ into PLINQ	230
ParallelEnumerable and Its AsParallel Method	232
AsOrdered and the orderby Clause	233
Specifying the Execution Mode	237
Understanding Partitioning in PLINQ	237
Performing Reduction Operations with PLINQ	242
Creating Custom PLINQ Aggregate Functions	245
Concurrent PLINQ Tasks	249
Cancelling PLINQ	253
Specifying the Desired Degree of Parallelism	255
WithDegreeOfParallelism	255
Measuring Scalability	257
Working with ForAll	259
Differences Between foreach and ForAll	261
Measuring Scalability	261
Configuring How Results Are Returned by Using WithMergeOptions	264
Handling Exceptions Thrown by PLINQ	266
Using PLINQ to Execute MapReduce Algorithms	268
Designing Serial Stages Using PLINQ	271
Locating Processing Bottlenecks	273
Summary	273

CHAPTER 7: VISUAL STUDIO 2010 TASK DEBUGGING CAPABILITIES 275

Taking Advantage of Multi-Monitor Support	275
Understanding the Parallel Tasks Debugger Window	279
Viewing the Parallel Stacks Diagram	286
Following the Concurrent Code	294
Debugging Anonymous Methods	304
Viewing Methods	305
Viewing Threads in the Source Code	307
Detecting Deadlocks	310
Summary	316

CHAPTER 8: THREAD POOLS 317

Going Downstairs from the Tasks Floor	317
Understanding the New CLR 4 Thread Pool Engine	319
Understanding Global Queues	319
Waiting for Worker Threads to Finish Their Work	329
Tracking a Dynamic Number of Worker Threads	336
Using Tasks Instead of Threads to Queue Jobs	340
Understanding the Relationship Between Tasks and the Thread Pool	343
Understanding Local Queues and the Work-Stealing Algorithm	347
Specifying a Custom Task Scheduler	353
Summary	359

CHAPTER 9: ASYNCHRONOUS PROGRAMMING MODEL 361

Mixing Asynchronous Programming with Tasks	362
Working with TaskFactory.FromAsync	363
Programming Continuations After Asynchronous Methods End	368
Combining Results from Multiple Concurrent Asynchronous Operations	369
Performing Asynchronous WPF UI Updates	371
Performing Asynchronous Windows Forms UI Updates	379
Creating Tasks that Perform EAP Operations	385
Working with TaskCompletionSource	394
Summary	398

CHAPTER 10: PARALLEL TESTING AND TUNING 399

Preparing Parallel Tests	399
Working with Performance Profiling Features	404
Measuring Concurrency	406

Solutions to Common Patterns	416
Serialized Execution	416
Lock Contention	419
Lock Convoys	420
Oversubscription	423
Undersubscription	426
Partitioning Problems	428
Workstation Garbage-Collection Overhead	431
Working with the Server Garbage Collector	434
I/O Bottlenecks	434
Main Thread Overload	435
Understanding False Sharing	438
Summary	441

**CHAPTER 11: VECTORIZATION, SIMD INSTRUCTIONS, AND
ADDITIONAL PARALLEL LIBRARIES** **443**

Understanding SIMD and Vectorization	443
From MMX to SSE4.x and AVX	446
Using the Intel Math Kernel Library	447
Working with Multicore-Ready, Highly Optimized Software Functions	455
Mixing Task-Based Programming with External Optimized Libraries	456
Generating Pseudo-Random Numbers in Parallel	457
Using Intel Integrated Performance Primitives	461
Summary	468

APPENDIX A: .NET 4 PARALLELISM CLASS DIAGRAMS **469**

Task Parallel Library	469
System.Threading.Tasks.Parallel Classes and Structures	469
Task Classes, Enumerations, and Exceptions	471
Data Structures for Coordination in Parallel Programming	472
Concurrent Collection Classes: System.Collections.Concurrent	474
Lightweight Synchronization Primitives	476
Lazy Initialization Classes	477
PLINQ	477
Threading	479
Thread and ThreadPool Classes and Their Exceptions	479
Signaling Classes	479
Threading Structures, Delegates, and Enumerations	480
BackgroundWorker Component	486

APPENDIX B: CONCURRENT UML MODELS	487
Structure Diagrams	487
Class Diagram	487
Component Diagram	489
Deployment Diagram	489
Package Diagram	489
Behavior Diagrams	489
Activity Diagram	491
Use Case Diagram	491
Interaction Diagrams	493
Interaction Overview Diagram	493
Sequence Diagram	494
APPENDIX C: PARALLEL EXTENSIONS EXTRAS	497
Inspecting Parallel Extensions Extras	497
Coordination Data Structures	502
Extensions	507
Parallel Algorithms	513
Partitioners	516
Task Schedulers	517
INDEX	521