

IMPLEMENTING PARALLEL PATTERNS IN .NET

Reed Copsey, Jr.

C Tech Development Corporation

Blog - <http://reedcopsey.com>

Twitter - <http://twitter.com/ReedCopsey>

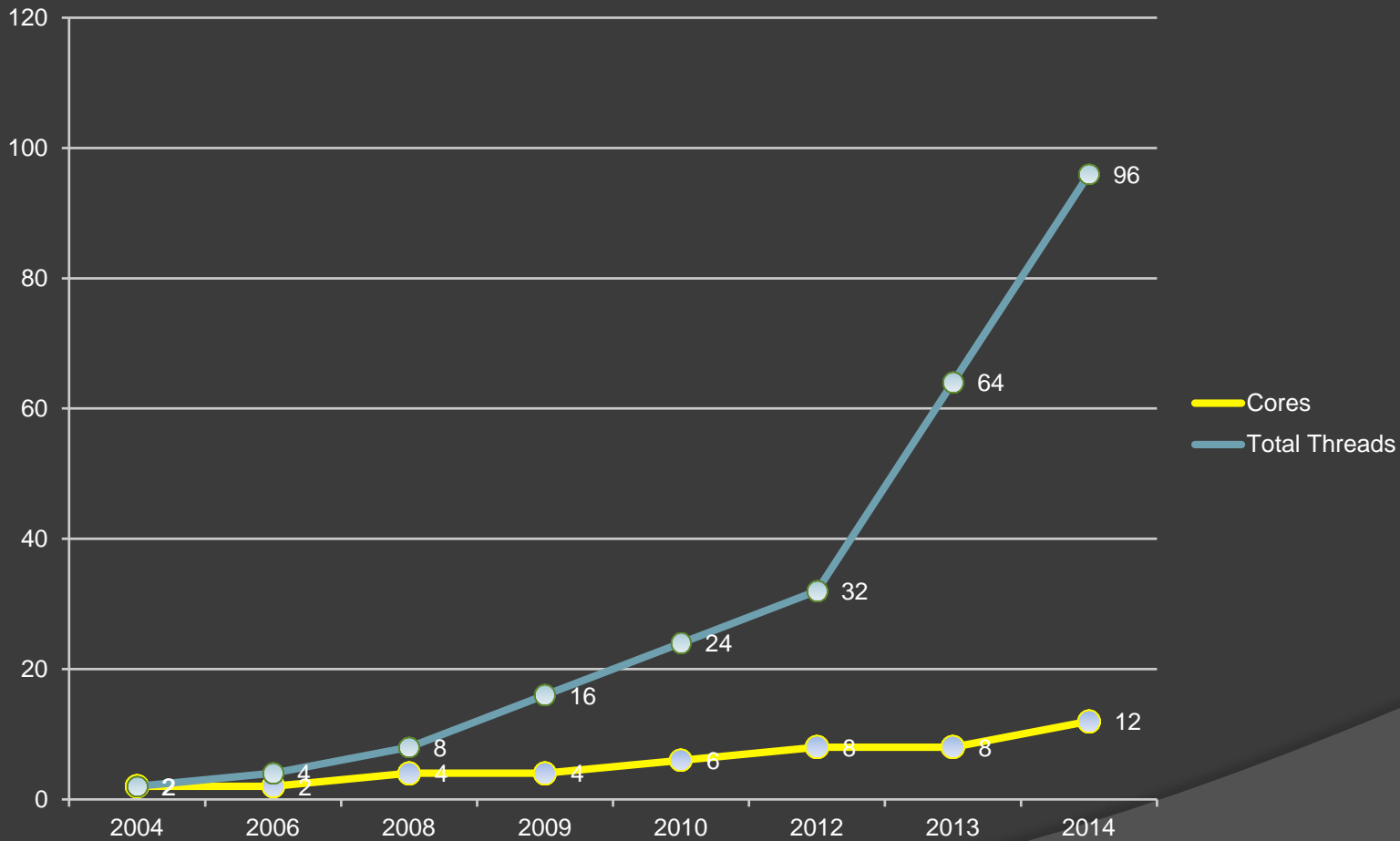
Why Parallelism is Important – Moore's Law

- ⦿ In 1965 Gordon Moore predicted that “the number of transistors on a chip will double every 24 months.”
- ⦿ The free lunch is over
 - No more doubling of CPU frequency
 - More work = Upgraded Computer
- ⦿ Moore's Law now applies to CPU cores

Trends in Computers

- 2006 – Dual Core
- 2007 – Quad Core
- 2008 – 2x Quad Core
- 2009 – 2x Dual and Quad Core with Hyper-Threading (2 Threads/Core)
- 2010 – 2x Six Core Systems with Hyper-Threading (24 Hardware Threads!)
- 2011+ – More Cores, with 4 and 8 Threads per Core!

Hardware Threads over Time



DEMO:

WHY PARALLEL?

Design Patterns for Parallelism

- ⦿ Design Pattern
 - “A reusable solution to a commonly occurring problem”
- ⦿ Common Language and Jargon Required
- ⦿ Applies to Parallelism

Common Terms for Parallelism

- ⦿ Task
- ⦿ Unit of Execution (UE)
- ⦿ Synchronization
- ⦿ Common Errors:
 - Race Conditions – Ordering matters
 - Deadlock

Steps to Parallel Programming

1. Find Concurrent Tasks
2. Understand Dependencies
3. Group Tasks to Simplify
4. Implement Using Patterns

Finding Concurrent Tasks

- Data Parallelism
- Task Parallelism
- Asynchronous Programming Patterns

Imperative Data Parallelism

◎ .NET 2.0/3.5

- ThreadPool via QueueUserWorkItem
- Self-Managed
- Difficult to do well

◎ .NET 4.0

- Parallel Class (System.Threading.Tasks)
 - Smart Partitioning
- Improved ThreadPool
 - Work Stealing
 - Hill Climbing

Effective, but Difficult.

More effective, and easier.

DEMO:

IMPERATIVE DATA PARALLELISM

Declarative Data Parallelism

● PLINQ

- Supports all LINQ to Objects syntax
- `IEnumerable<T> -> ParallelQuery<T>`
- `.AsParallel()`

DEMO:

DECLARATIVE DATA
PARALLELISM

Task Parallelism

◎ .NET 2.0/3.5

- Thread Class
- ThreadPool
- BackgroundWorker
- Synchronization via locking

◎ .NET 4.0

- Parallel.Invoke
- Task class
 - Continuations
 - Cancellation
 - AggregateException
 - Child Tasks
- Parallel Stacks Debugging

Common Issues

- ⦿ Beware the UI thread
 - SynchronizationContext for portability
 - Use context.Post and context.Send instead of Control.BeginInvoke() and Dispatcher.Invoke()
- ⦿ Exception Handling
 - Debugging difficulties
- ⦿ Synchronization
- ⦿ Sharing Effectively

DEMO:

TASK PARALLELISM

Asynchronous Programming Patterns

◎ .NET 2.0/3.5

- IAsyncResult
 - Delegate.BeginInvoke
- IAsyncResult
 - BeginXXX() and EndXXX()
 - Example: WebRequest - BeginGetResponse() and EndGetResponse()

◎ .NET 4

- Use Task / Task<T> directly on delegate
- TaskFactory.FromAsync
 - Wrap IAsyncResult method
- Fork/Join
 - Task.Invoke
 - Task.StartNew() with Task.Result

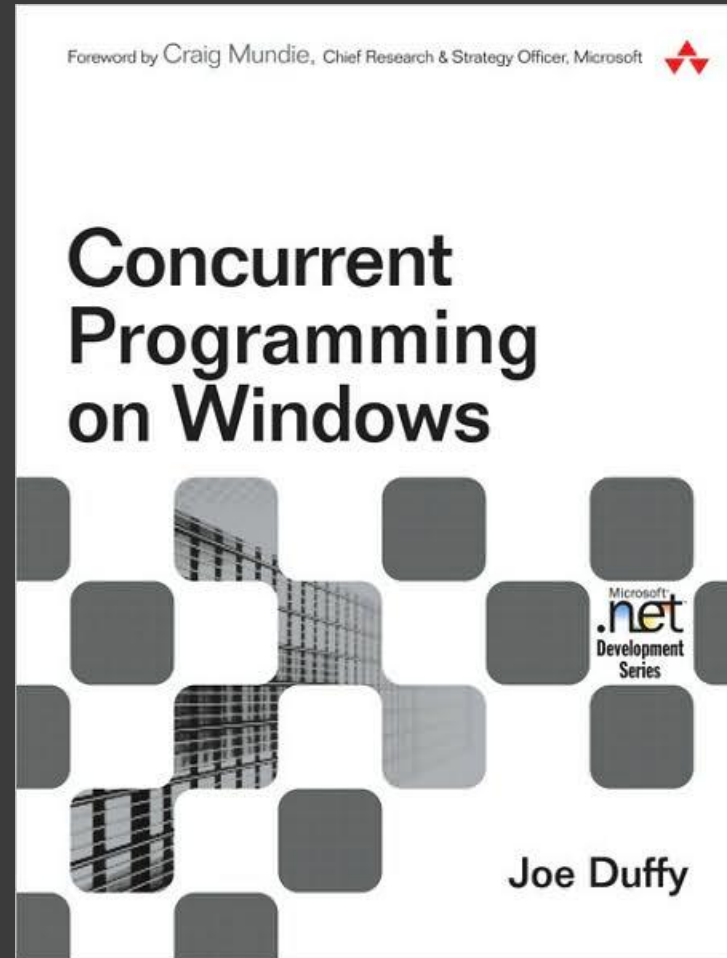
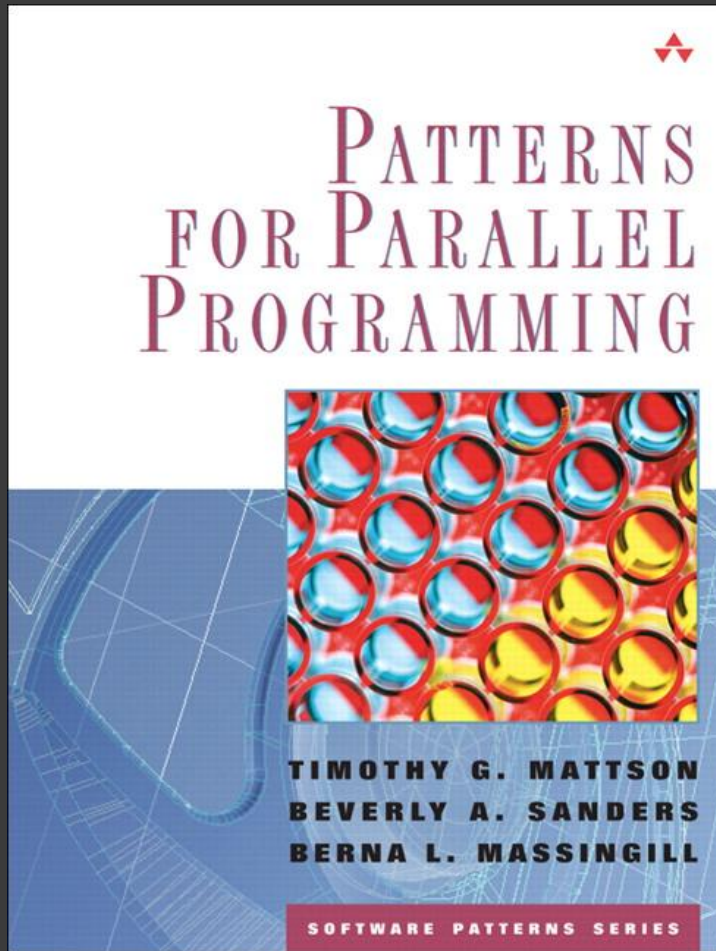
Common Threading Issues

- ◎ Closures
- ◎ True Sharing
- ◎ False Sharing
 - 64 or 128 byte Cache Lines
- ◎ Memory Allocations in Delegates
 - Algorithm is only as scalable as the GC
 - Server GC (throughput) vs. Workstation GC (latency)

DEMO:

COMMON PROBLEMS

Books for More Information



Online Resources

- ◎ Parallel Computing Developer Center
 - <http://msdn.microsoft.com/en-us/concurrency/default.aspx>
 - Microsoft Parallel Team Blogs
- ◎ “Patterns of Parallel Programming” article by Stephen Toub
- ◎ My Blog: <http://reedcopsey.com>