



An Introduction to What's Available in .NET 4.0

PARALLEL PROGRAMMING ON THE .NET PLATFORM

About Me



- Jeff Barnes
- .NET Software Craftsman @ DAXKO
- Microsoft MVP in Connected Systems
- ALT.NET Supporter

- jeff@jeffbarnes.net
- <http://jeffbarnes.net/blog>
- http://twitter.com/jeff_barnes






Disclaimer

I do not claim to be an expert in anything!


I'm here to simply share things that I have learned regarding topics in which I am quite passionate.



Question everything I say and form your own opinions based on the information!



Agenda

- What is parallel programming?
 - Why is it necessary?
 - Why is it so hard (or perceived to be)?
 - What can .NET 4.0 do to make it easier?
- 

What is Parallel Programming?

- Basically means “multi-threading” or doing more than one thing at the same time
- Often refers to:
 - Various patterns to achieve parallelism in different scenarios.
 - Higher level abstractions than simply working with threads.



Do I Really Need To Use It?


In a word...

YES!!!





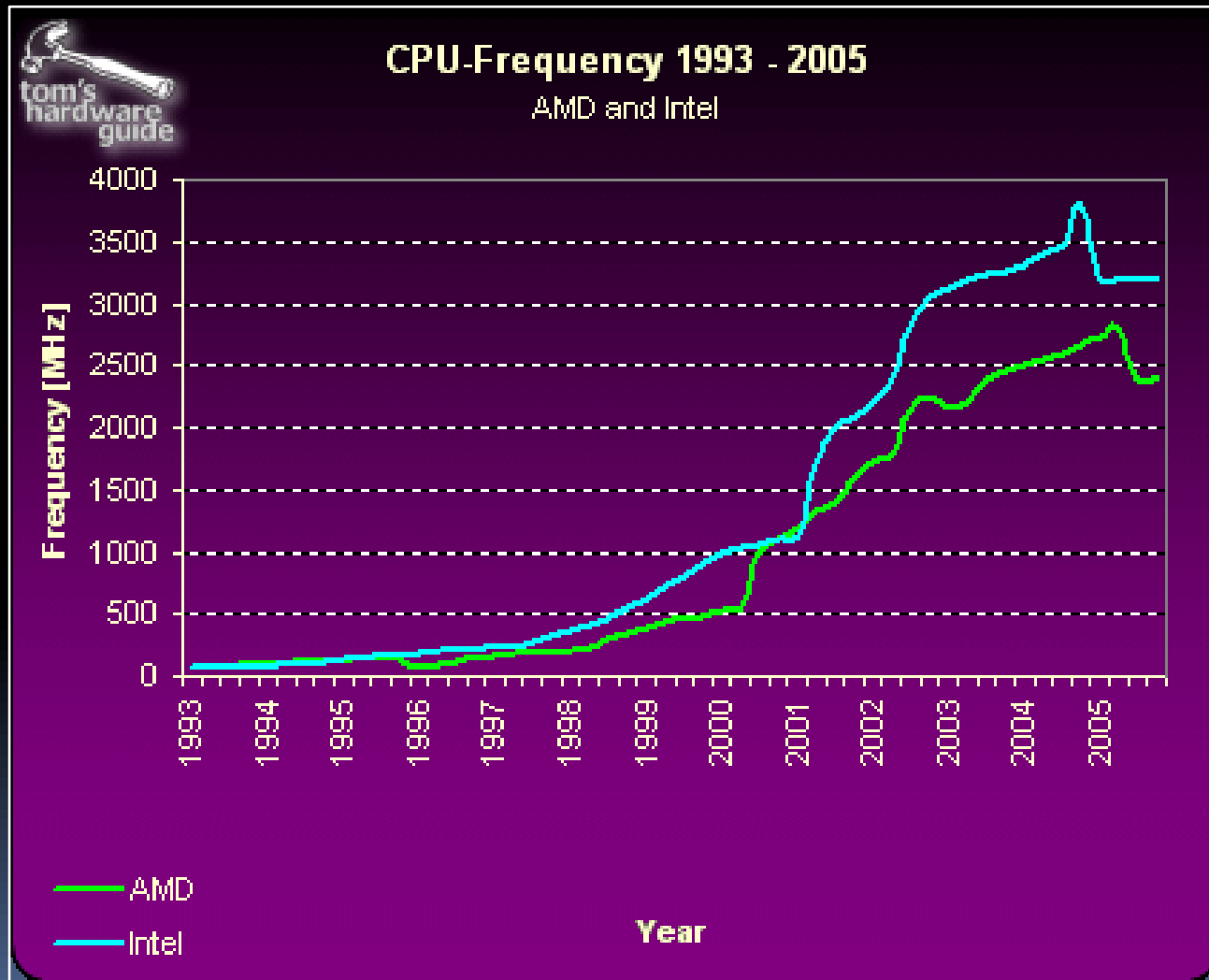
But...Be Practical

- Not literally every app requires it, but the potential benefit frequently exists.
 - Current hardware will increasingly necessitate parallelism to achieve desired levels of throughput and scalability
- 

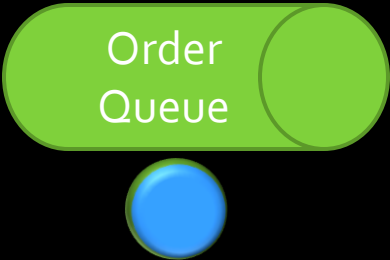
Hardware Limitations

- CPUs aren't getting "faster"
- Cores will only increase in the years to come
- The Result:
 - Can't get away with waiting for CPU to get faster
 - Serial (traditional) logic won't scale to multi-cores
 - Parallelism will no longer be an option

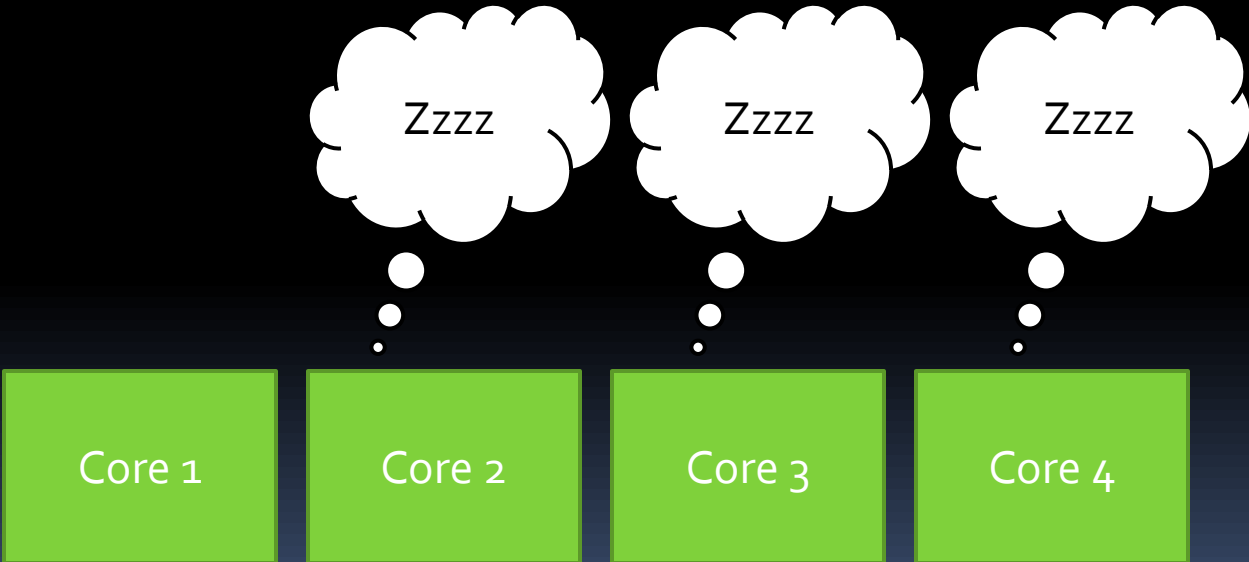
History of Clock Rate



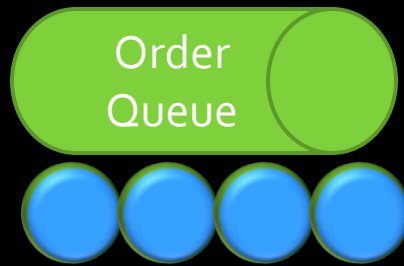
Serial Code on Multi-Cores



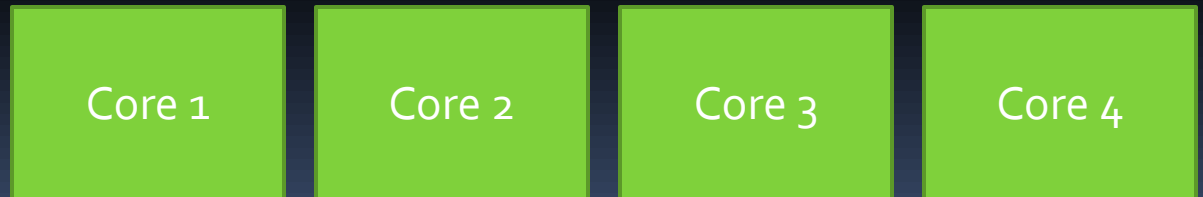
```
foreach (var order in get_orders_from_queue())  
{  
    process(order);  
}
```



Parallel Code on Multi-Cores



```
foreach (var order in get_orders_from_queue())  
{  
    // Asynchronous Dispatching Goo Goes Here  
    process(order);  
}
```




Why Is Parallelism So Hard?

- Threads can be difficult to grok
- Mutable state shared between threads
- Synchronization is complex and error prone
- Increases level of programming complexity
- Simply put: it is not a trivial problem area




Subtle Problems Can Bite You

- Increment / Decrement Count
 - Dictionary Operations
 - Add To or Removing From a List

 - Let's look at an example
- 




.NET 4.0 To The Rescue

- Concurrent Collection Data Structures
 - New Lock Primitives
 - Overhaul of the Thread Pool
 - Parallel LINQ
 - Task Parallel Library
 - Debugging Support in VS2010
 - Profiling Support in VS2010
- 



Concurrent Data Structures

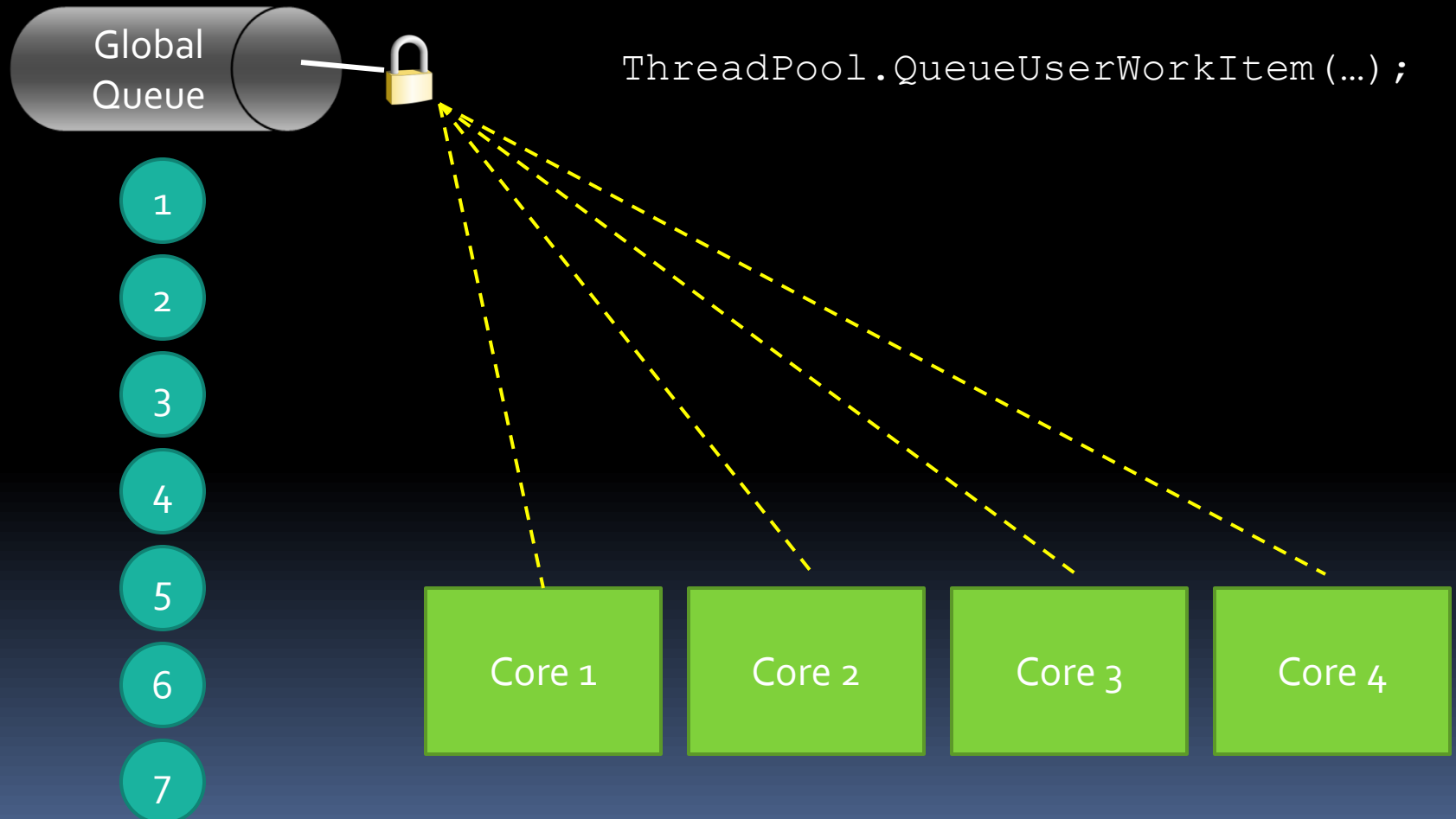
- Found in `System.Collections.Concurrent`
 - `BlockingCollection`
 - `ConcurrentBag`
 - `ConcurrentDictionary`
 - `ConcurrentQueue`
 - `ConcurrentStack`
- 



Locking Primitives

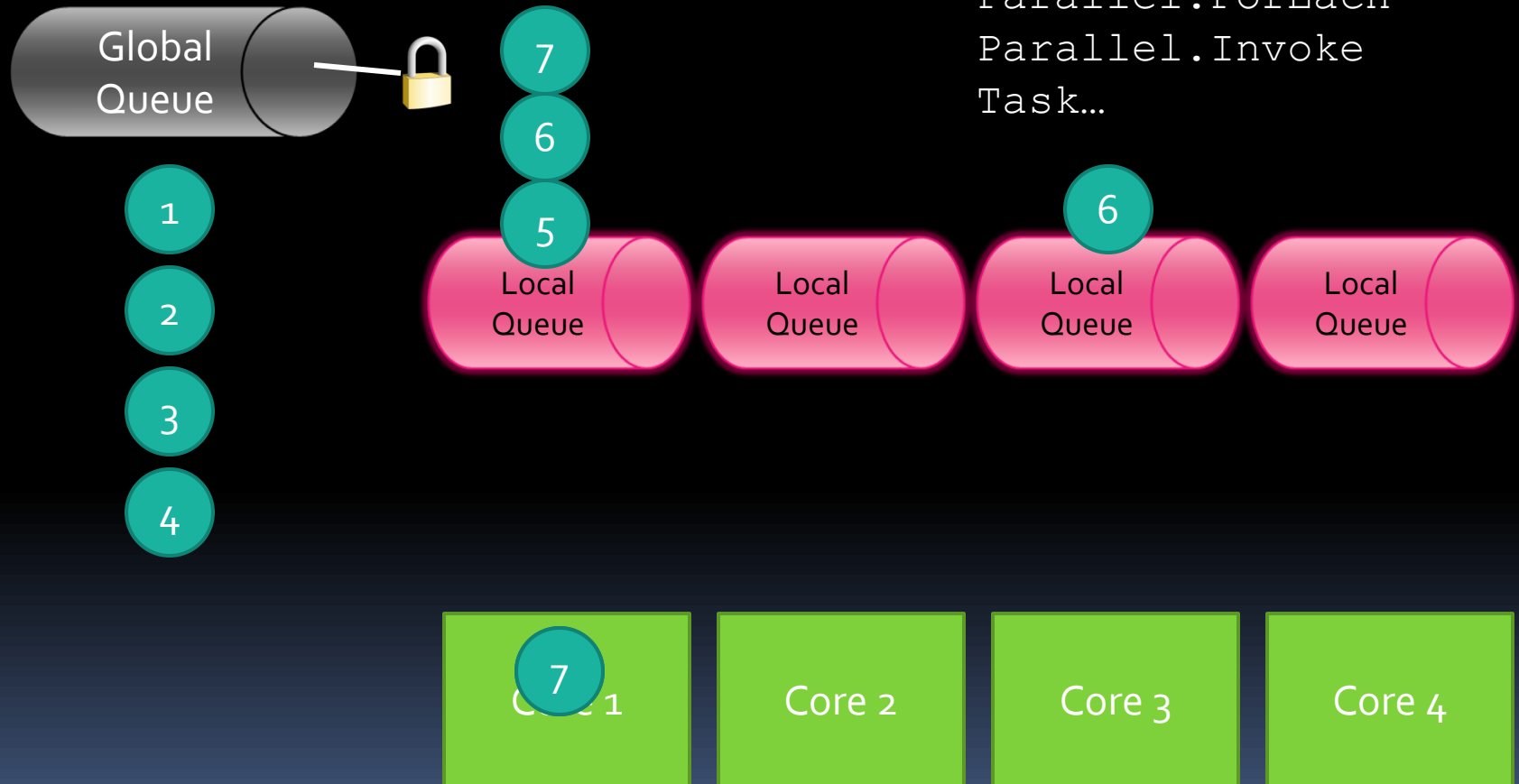
- SemaphoreSlim
 - SpinWait
 - SpinLock
 - CountdownEvent
 - ManualResetEventSlim
- 

.NET 3.5 Thread Pool




.NET 4.0 Thread Pool

```
Parallel.For  
Parallel.ForEach  
Parallel.Invoke  
Task...
```





Parallel Extension Methods

- Enables you to easily parallelize iterative blocks of code
 - `Parallel.For`
 - `Parallel.ForEach`
 - `Parallel.Invoke`
- 



Parallel LINQ

- Made possible via `IParallelEnumerable`
 - Inherits `IEnumerable`
- Adds a few methods (`AsParallel`)





Lazy Initialization

- Lazy<>
- Allows first class support for lazily initializing a value upon the first request.
- Much easier than hand rolling your own solution.




Tasks

- Task represents an asynchronous operation
 - Higher level abstraction than a thread
 - Much easier to work with
 - Supports concepts such as:
 - Continuations
 - Tracking Status
 - Cancellation




VS2010 Profiling & Debugging

- Tasks Window
 - Threads Window
 - Profiling for:
 - CPU
 - Memory
 - Concurrency
 - And More
- 



Key Takeaways

- .NET 4.0 makes parallelism much easier
 - Not a silver bullet
 - You still have to be aware of common issues
 - Be mindful of shared state!
- 





Resources

- <http://msdn.microsoft.com/concurrency>
 - From there, you can find:
 - Links to MSDN Code Gallery
 - Link to Stephen Toub's Excellent Paper
 - Links to other pertinent blogs and samples
- 