Sela.

DNParallel

Parallel Programming and TPL Dataflow

college@sela.co.il 03-6176666

Parallel Programming and TPL Dataflow

DNParallel - Version: 3

4 days Course

Description:

In this course, you will learn how to use the new .NET 4+ features to parallelize existing code, to utilize data parallelism, to create and manage tasks, and to synchronize access to data using concurrent collections.
You will gain deeper understanding on Task, Async/Await, TPL Dataflow.
The course will discuss best practice, api design and guidelines.

Intended audience:

.NET developers with at least 1 year of C# programming experience. Cursory familiarity with operating system concepts such as multithreading and synchronization is recommended.

Prerequisites:

Objectives:

You will gain deeper understanding on Task, Async/Await, Tpl Dataflow.
The course will discuss best practice, api design and guidelines
Understand TPL Dataflow blocks and performance tuning

Topics:

Parallel from start

- Intoduction
  - History
  - Why Do We Need Parallelism?
  - Threads
  - The Downside of Parallelism
  - .NET 1
  - Thread
  - The ThreadPool
  - Thread vs Thread Pool
- Sync
  - Amdahl's Law
  - How to Parallelize a Unit of Work?
  - Stack and Heap
  - Thread safety
  - Critical Sections (Lock)
  - Lock vs. Interlocked
  - CountdownEvent
  - Quiz
- Collections
  - Concurrent Collections
  - ConcurrentQueue<T>
  - ConcurrentStack<T>
  - ConcurrentBag<T>
  - IProducerConsumerCollection<T>
  - BlockingCollection<T>
  - ConcurrentDictionary
  - Other APIs
- Task
  - Explicit Parallelism

- º Task == Data Structure
- º Task<T>
- º Task as data structure
- º Semantical Task
- º Scheduling Task using Factory
- º Long-Running Tasks
- º Thread Safety

• Continuation
- º A better callback API
- º Tasks vs. APM
- º Continuations
- º WhenAll, WhenAny
- º Conditional Continuations
- º Continuation Configurations

• Async
- º Is it the best we can do?
- º C# 5: async syntax
- º Chaining await
- º Simultaneous await
- º When All
- º Using

• Loops
- º Implicit Parallelism
- º Async Loop Strategies
- º Await loop
- º Await LINQ Pattern

• IOCP
- º IO operations
- º Service call without IOCP
- º IOCP
- º Async Download

    - TAP: adoption
    - Task base WCF proxy / server side by default
    - Task base Web API
- Diagnostic
    - Debugger Parallel Watch window
    - Parallel Stacks
- Exceptions
    - Handling Exception
    - Exception flow control
    - AggregateException
    - Catching aggregate exceptions
    - Read-able exceptions
- Cancellation
    - Cancellation
    - Polling for Cancellation
    - Cancellation and Tasks
    - Timeouts and Cancellation
- Summary

TPL Dataflow

- What is TPL Dataflow?
    - The agent base concept
    - Evolution
    - Goals
- Getting started
    - Namespace
    - NuGet
- Contract
    - Source API
    - Target API

    - Block API
    - Push vs. Pool
- Blocks
    - Blocks categories
- Action block
    - Structure
    - Functionality
    - Throttling
- Buffer Block
    - Structure
    - Functionality
    - Push and Pool
    - Bounded capacity
- Broadcast Block
    - Structure
    - functionality
    - What makes it different than Buffer Block?
- Transform Block
    - Structure
    - Functionality
- Transform Many Block
    - Structure
    - Functionality
- TPL Dataflow and Async
    - Using async / await with TDF
    - Processing I/O operations
- Performance tuning
    - MaxMessagesPerTask
- Case Study: Web Crawler
- Appendixs:
    - Batch Block

    - Join Block
    - Greediness
    - Greediness and built-in blocks
    - BatchedJoinBlock
    - WriteOnceBlock
    - Rx vs. TDF
- Summary