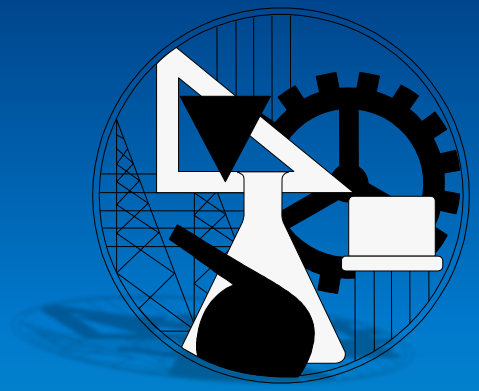


INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Engenharia Informática e de Computadores

Programação Paralela na.NET Framework 4.0

Apresentação individual de Projecto e Seminário 2009/2010

Ricardo Neto
26657@alunos.isel.pt

CONTEXTO

"O número de transis… …licará por cada 2 anos"

Problemas de Temperatura e consumo!

Aumento do número de *cores* em vez do *clock cycle*.

- **O que temos vindo a aprender...**
  - Criação/destruição de threads
  - Utilização de primitivas de sincronização
  - Manipulação da Threadpool (via utilização QueueUserWorkItem)
- **O que é disponibilizado na .NET Framework 4.0...**
  - Bibliotecas que visam simplificar o desenvolvimento paralelo
  - Formas de evitar manipulação directa de Threads e da Threadpool
  - Estado de execução e contrução parametrizada de um item de trabalho

Segundo Joe Duffy:
A criação de uma thread custa 200K ciclos CPU e a sua destruição 100K!
Um context switch custa entre 2K a 8K ciclos!

Threadpool .NET 3.5

Serviço, disponibilizado pelo CLR que, guarda items de trabalho a serem executados por worker threads.

- **Funcionamento**

  - Inicia sem worker threads e cria as necessárias para atender pedidos até ao número mínimo definido na configuração.

  - Quando detecta threads em execução num número igual ou superior ao mínimo cria novas threads (1 thread/0,5 s.).

  - O limite máximo de threads é também um parâmetro configurável.

  - Após a execução de um item de trabalho, as worker threads bloqueiam-se aguardando um novo item de trabalho.

  - Após 10 segundos sem item de trabalho são destruídas.

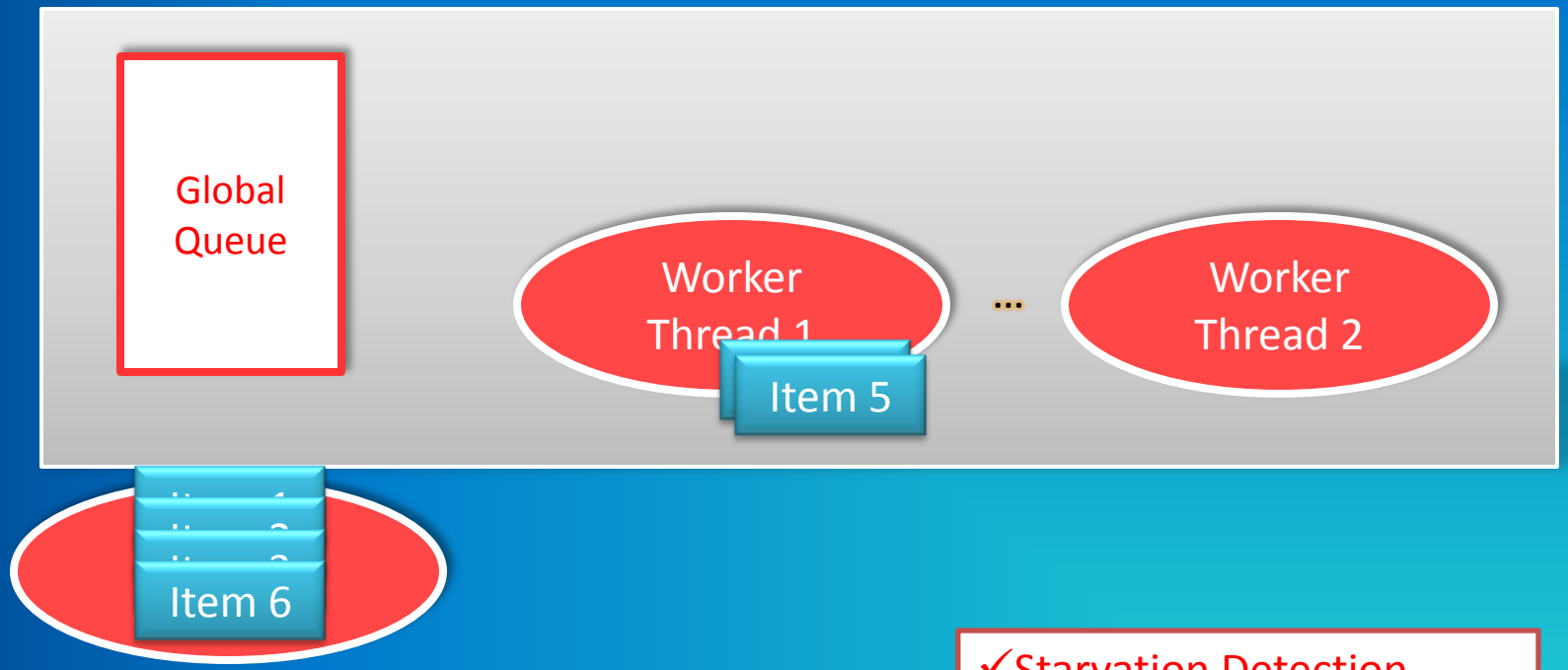
- **Fila Global de Items de Trabalho**

  - Versão 2.0: A mesma fila global para todos os AppsDomains.

  - Versão 3.5: Uma fila por AppDomain.

  - Em ambas as versões lock global a progeter as operações de Push e Pull.

Threadpool .NET 3.5



- ✓Starvation Detection
- ✓ Idle Thread Retirement

"Microsoft Parallel Programming With Visual Studio 2010 and the .NET Framwork 4"
Toub, Stephen – Microsoft Corporation

Threadpool .NET 4.0

Num futuro muito próximo máquinas com 16 processadores serão comuns

- **Push/Pull de work items**

  - Alteração da fila global passando a implementa-la com uma ConcurrentQueue<T> em vez da LinkedList utilizada anteriormente.

  - A fila comporta-se como um array com um ponteiro afectado atomicamente.

- **Worker Thread Queues e Work Stealing**

  - Cada worker thread tem uma fila própria de items de trabalho.

  - Uma thread idle procurará executar items de trabalho presentes nas filas de outras worker threads.

  - Esta feature só é aplicada à utilização da TPL.

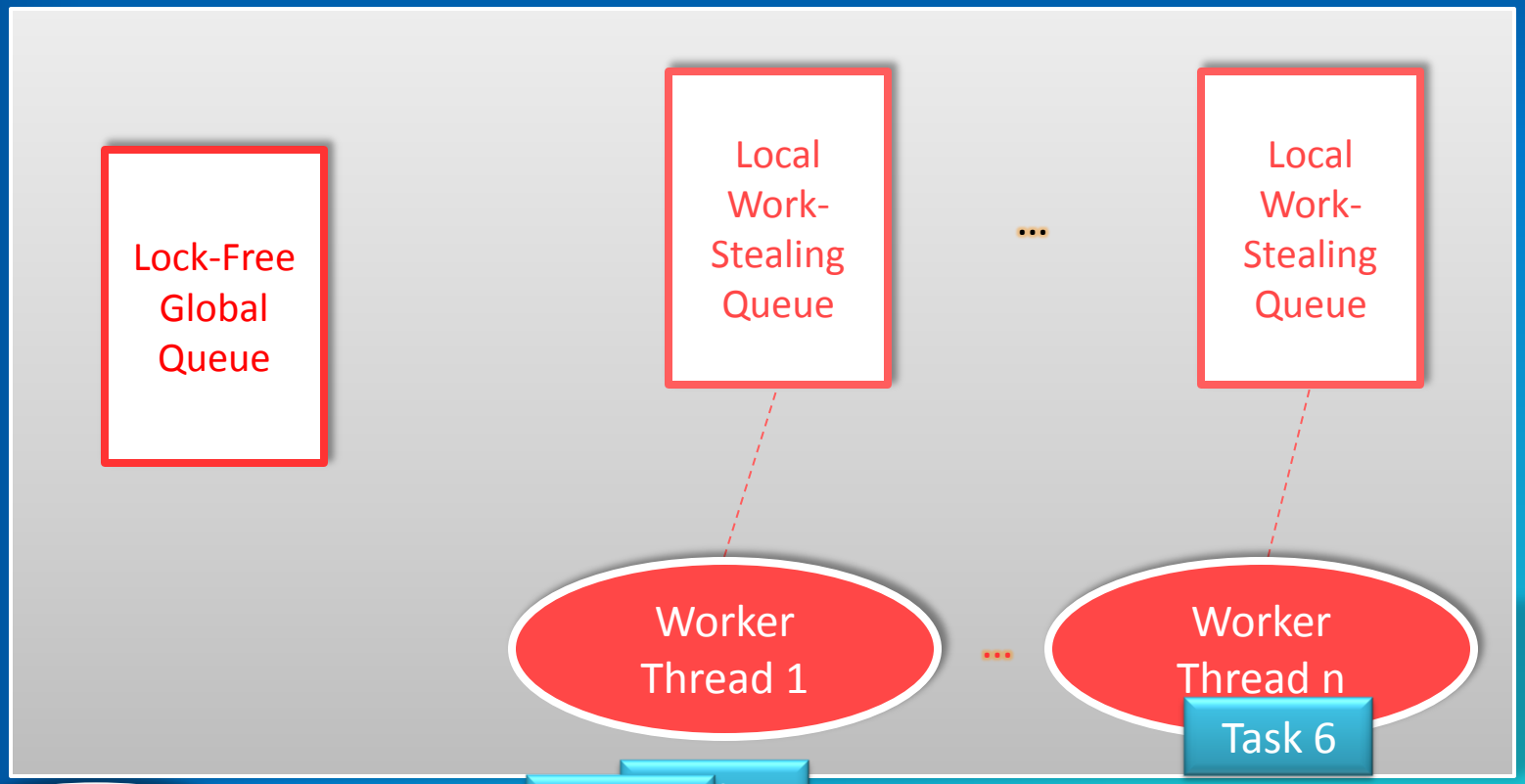
- **Diminuição da Concorrência**

  - Adiamento de execução de items de trabalho com vista à redução da concorrência.

- **Número máximo de worker threads**

  - O número máximo de threads passa a depender da memória disponível no espaço de endereçamento virtual do processo.

Threadpool .NET 4.0



- ✓Starvation Detection
- ✓ Idle Thread Retirement
- ✓Hill-climbing

"Microsoft Parallel Programming With Visual Studio 2010 and the .NET Framwork 4"
Toub, Stephen – Microsoft Corporation

Threading/Concorrência vs. Paralelismo

A primeira ideia que nos ocorre quando falamos em paralelismo é threads.

- **Concorrência**

  - Threads tipicamente utilizadas para optimizar o tempo de resposta de uma aplicação (não bloqueamento da UI, chamadas assíncronas,...).

  - A concorrência permite assim tentar fazer a melhor utilização do CPU na presença de várias tarefas.

  - Nestes cenários, o tópico que implica alguma atenção é o sincronismo.

- **Paralelismo**

  - Forma de optimizar operações compute-bound, tirando partido de ambientes multi-core.

  - Uma das utilizações mais comum é o particionamento de um processo por vários fios de execução, executando-os em simultâneo.

Podemos ter concorrência numa arquitectura com um ou mais cores mas só podemos ter paralelismo numa arquitectura multi-core.

Principio para Paralelização: Decomposição

- **Decomposição de Dados**

  - Como forma de resolver um problema de forma paralela, deve-se dividir o dominio, permitindo que várias partes sejam tratadas em simultâneo.

  - Tipicamente um processamento sobre cada item de uma colecção é um bom candidato a paralelização (e.g. for, foreach...)

Quando o processamento de cada elemento é dependente de outro a paralelização pode afectar a performance!

Exemplo: Modificação do contraste de uma imagem

- **Decomposição de Acções**

  - Divisão de um algoritmo em acções que podem ser realizadas em simultâneo

  - A divisão de tarefas implica a análise de um algoritmo e a forma como realiza as suas acções.

Exemplo: Escrever dados de formulário numa BD e enviar e-mail

.NET 4.0 - Task Parallel Library

- Conjunto de tipos e API's que visam facilitar a implementação de aplicações que envolvam paralelismo e concorrência.

- Gestão do nível de concorrência dinâmicamente por forma a optimizar a utilização dos CPU's existentes.

- Oferece decomposição automática de operações, esclonamento de tarefas na Threadpool, suporte para cancelamento, gestão de estado....

Programador com mais atenção ao objectivo da aplicação

- **User Mode Scheduling**

  - Mecanismo utilizado pelas aplicações para escalonarem as suas próprias threads em user-mode sem recorrerm ao kernel.

  - Mais eficiente que a Threadpool para gerir um grande número de work items de curta duração.

.NET 4.0 – TPL *(System.Threading.Tasks)*

- **Paralelismo de Dados (Data Parallelism)**

  - Suportado pelos métodos estáticos For e ForEach da classe System.Threading.Tasks.Parallel

```
for (int row = 0; row < image.GetUpperBound(0); row++)
{
   for (int col = 0; col < image.GetUpperBound(1); col++)
      image[row, col] = ContrastChanger(image[row, col]);
}
```

80863 ms.

```
Parallel.For(0, image.GetUpperBound(0), row =>
   {
      for (int col = 0; col < image.GetUpperBound(1); col++)
         image[row, col] = ContrastChanger(image[row, col]);
   });
```

14971 ms.

- 81%

  - O método Parallel.For retorna um objecto do tipo ParallelLoopResult

  - Este dá informações acerca da iteração mais baixa onde poderá ter ocorrido um break e se a execução foi até ao fim

.NET 4.0 – TPL *(System.Threading.Tasks)*

- **Paralelismo de Acções (Task Parallelism)**

  - Objecto Task que representa uma operação assíncrona.

  - Uma Task pode implicar a criação de uma Thread ou item de trabalho.

  - Permite abstracção sobre os recursos existentes e oferece uma API que suporta espera, cancelamento, continuação, tratamento de excepções,....

- **Criação e Execução implícita de Tasks**

```
SendEmail(address, msg);
PersistData(address, msg);
```

1995 ms.

```
Parallel.Invoke(() => SendEmail(address, msg),
 () => PersistData(address, msg));
```

1007 ms.

- 48%

- **Criação e Execução Explicita de Tasks**

```
var taskA = new Task(() => SendEmail(address, msg));
var taskB = new Task(() => PersistData(address, msg));
```

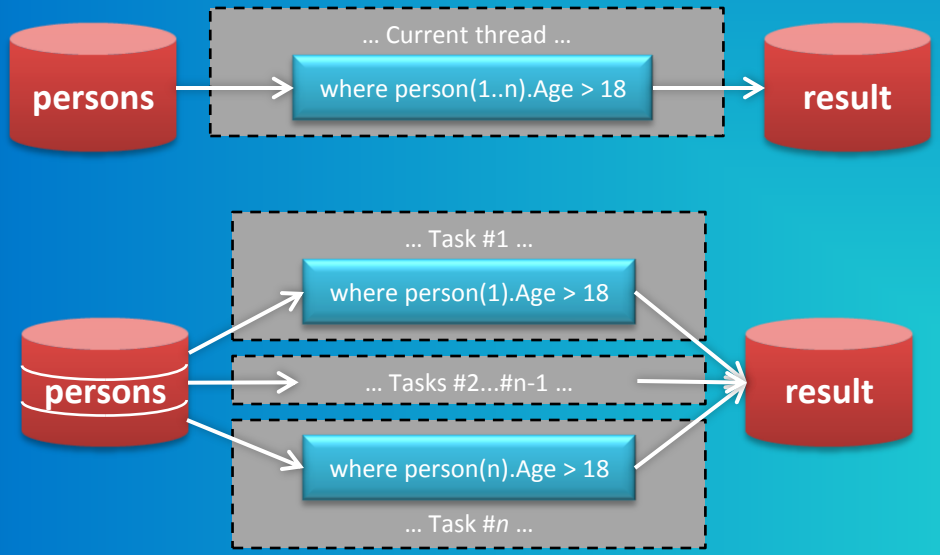
... Wait, WaitAll, WaitAny, Start, RunSynchronously, ContinueWith ...

.NET 4.0 – Parallel LINQ

- **O que é uma query paralela?**

  - O LINQ é uma biblioteca introduzida na versão 3.0 da Framework que disponibiliza métodos capazes de interrogar um IEnumerable<T>.
  - PLINQ é uma implementação de LINQ que permite a execução paralela de uma query, tirando melhor partido dos recursos existentes.

```
var result = from person in persons.AsParallel()
     where person.Age > 18
     select person;
```



"Microsoft Parallel Programming With Visual Studio 2010 and the .NET Framwork 4"
Toub, Stephen – Microsoft Corporation

.NET 4.0 – Estruturas de Dados

- **Colecções Concorrentes**

  - Classes de colecções que disponibilizam leituras e escritas thread-safe.
  - Permitem utilização sem preocupação com locks nos acessos

```
BlockingCollection<T>, ConcurrentBag<T>, ConcurrentQueue<T>,
ConcurrentDictionary<T>, ConcurrentStack<T>
```

- **Primitivas de Sincronização**

  - Primitivas que permitem concorrência fine-grained e melhores performance, evitando locks penalizadores

```
Barrier, CountdownEvent, ManualResetEventSlim, SemaphoreSlim,
SpinLock, SpinWait
```

- **Iniciadores Lazy de Objectos**

  - Conjunto de classes que permitem iniciação lazy de objectos.
  - Permitem também definir afinidade do objecto à thread.

```
Lazy<T>, ThreadLocal<T>, LazyInitializer
```

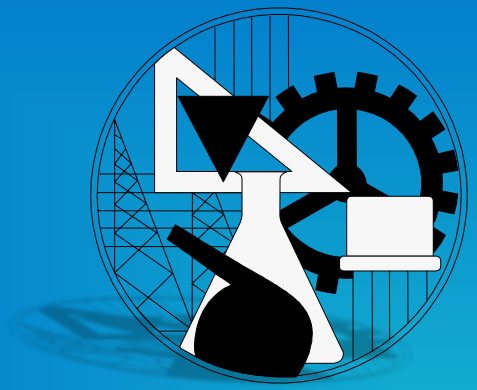
- **Excepções Agregadas**

  - Tipo utilizado para capturar múltiplas excepções que são lançadas em threads diferentes e são devolvidas à thread que jaz join.

Perguntas e Respostas

Referências Utilizadas

- **Microsoft Parallel Programming in the .NET Framework**
  http://msdn.microsoft.com/en-us/library/dd460693(v=VS.100).aspx
- **Microsoft Parallel Computing Developer Center**
  http://msdn.microsoft.com/en-us/concurrency/default.aspx
- **Microsoft Parallel Extensions Team Blog**
  http://blogs.msdn.com/pfxteam/
- **"*Design Codes*", Aviad Ezra Blog**
  http://aviadezra.blogspot.com/2009/04/task-parallel-library-parallel.html
- ***"Parallel Programming for Managed Developers with Visual Studio 2010"***
  Daniel Moth at Professional Developers Conference 2008
  http://channel9.msdn.com/pdc2008/TL26/



INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Engenharia Informática e de Computadores

Ricardo Neto

26657@alunos.isel.pt