

Jakub Čermák, Programátorské večery, MFF UK,
Praha 6.3.2009

PARALLEL EXTENSIONS TO THE .NET FRAMEWORK 3.5

Úvodem



- Knihovna pro .NET 3.5
- Zjednodušená práce s vlákny a více jádry čistě v managed kódu a pro managed kód
- Aktuálně CTP June 2008 (obsažena i ve .NET 4 CTP) + neveřejná beta verze
- Bude obsažena v .NET 4.0

- Proč paralelizovat? ... prostě protože vývoj směřuje k paralelizmu
- <http://blogs.msdn.com/pfxteam/>,
[http://msdn.microsoft.com/cs-cz/concurrency/default\(en-us\).aspx](http://msdn.microsoft.com/cs-cz/concurrency/default(en-us).aspx), ...



Úvodem (2)



- Doplňuje stávající podporu vláken v .NETu, nenahrazuje ji (!)
- Neřeší synchronizaci vláken, race-condition, locky atp. => je pořád přemýšlet nad (potenciálními) paralelizačními problémy – to za programátora nikdo nevyřeší
- „Pouze“ zjednodušuje některé případy paralelizace, kdy klasický (třída Thread) přístup není „názorný“



Struktura



- ◎ Task Parallel Library (TPL)
 - Imperativní paralelismus
 - Základní primitiva – Task
 - „vyšší“ funkcionalita – Parallel, Futures
 - Thread-safe kontejnery v `System.Threading.Collections`
- ◎ Parallel LINQ (PLINQ)
 - Rozšíření klasického LINQu o práci s více vlákny
 - Což díky jeho funkcionálním rysům jde pěkně
 - Deklarativní paralelismus



Třída Parallel



- Nejvyšší úroveň abstrakce
- 3 statické (přetížené funkce)
 - For – paralelní varianta For cyklu
 - ForEach – paralelní varianta foreach
 - Invoke – spuštění několika nezávislých fcí najednou
- Čeká na ukončení všech iterací
- Díky λ -výrazům vizuální podobnost k klasickými for/foreach cykly
- ParallelState – thread-local úložiště + zastavení výpočtu



Třída Parallel



- Demo ForExample



Třídy Future a LazyInit



- Vykonání akcí, jehož výsledek nechci hned
- Po vytvoření instance na nic nečeká a pokračuje vykonávání kódu a
 - Future: na pozadí se spustí vlákno s výpočtem – počká na dopočet výsledku (pokud ještě neskončil) a vrátí výsledek
 - LazyInit: nic se neděje – výsledek se začne počítat, až když si o něj řeknu
- `Future<int> answer = Future.Create<int>(UltimateAnswer);`
- `LazyInit<EncapsulatedInt> answer = new LazyInit<EncapsulatedInt>(UltimateAnswer2);`



Třída Task



- ⦿ Nejnižší úroveň abstrakce
- ⦿ Reprezentuje asynchroně vykonávanou fci
- ⦿ Základ pro „vyšší“ třídy (Parallel,...)
 - Agregace výjimek
 - Přerušování výpočtu a čekání na něj
 - Více násobné spuštění
 - ...



Ovlivňování paralelizace



⦿ TaskManager

- „Manažer“ a plánovač úloh (Tasků)
- Každý Task spravován nějakým TaskManagerem
- TaskManagerPolicy
 - Politika paralelizace
 - Kolik jader použít, kolik vláken na jádro,...

⦿ ParallelOptions

- Přidána až v Betě 1
- Přináší další možnosti nastavení pro jednotlivá primitiva



Třídy Future a LazyInit



- Demo FutureNTaskExample



Srovnání - OpenMP



- Cross-platform rozšíření kompilátoru C/C++
- Podpora ve Visual Studiu, gcc, Intel C Compiler, ...
- Ovládaná pomocí #pragma direktiv + API funkce pro nastavování, lockování, ...
- Obsahuje funkcionalitu ekviv. Parallel.For, Task + synchronizační primitiva a pomocné fce
- Lepší podpora sdílení proměnných
- Vyžaduje podporu v kompilátoru
- ...

```
#pragma omp parallel for schedule(static) reduction(+:MonteCarloPoints)
shared(pValues)
for(int DetRotAngleId=0; DetRotAngleId<DetRot_StepsCnt; ++DetRotAngleId)
{
    //nějaký výpočet
}
```



Parallel LINQ



- LINQ – Language Integrated Query
 - Přidává do .NETu způsob pro dotazování nad daty (IEnumerable, SQL2LINQ, XML, ...)
- PLINQ – rozšíření původních LINQovských rozhraní o paralelní přístup k datům



ParallelEnumerable, IParallelEnumerable



- Paralelní varianta klasických IEnumerable a Enumerable
- Obsahuje stejné (na 1. pohled metody), akorát paralelní
- Používají se stejně + hlídat race-condition, deadlocky, ...
- Získání rozhraní:
 - `public static IParallelEnumerable<T>
AsParallel<T>(this IEnumerable<T> source)`
 - `public static IEnumerable<T>
AsSequential<T>(this
IParallelEnumerable<T> source)`



Použití



```
var r = from i in Enumerable.Range(0, 12345)
        where IndecisiveSubject.Likes(i)
        select i;
```



```
var r = from i in Enumerable.Range(0, 12345).AsParallel()
        where IndecisiveSubject.Likes(i)
        select i;
```



PLINQ



- Dema plinq_example a RayTracing



◉ Děkuji za pozornost

