# Paralelní programování

přednášky

*Jan Outrata*

únor–květen 2011

# Simulátor konkurence

- abstrakce = libovolné proložení atom. akcí sekvenčních procesů
- konkurentní program **nelze klasicky ladit** – při každém spuštění může být jiné proložení = jiný scénář výpočtu

= **interpret konkurentního programu umožňující** uživateli **kontrolovat proložení atom. akcí** procesů

- po každé atom. akci zobrazí stav programu a volitelně program pozastaví a umožní výběr následující akce a z následujících akcí procesů, tzn. výběr procesu

= paralelní architektura emulující konkurentní prostředí – přepínáním samostatných kontextů procesů

- př. **Spin** – také pro ověření korektnosti programu, tzv. **model checker**
- př. **BACI (Ben-Ari Concurrency Interpreter)** – výukový, překladač a interpret dialektů jazyků C a Pascal, konstrukty **cobegin** (konkurentní vykonávání procedur, implicitní bariéra) a **delay**

# Model checker

- dokazování korektnosti je možné pomocí stavového diagramu
- **stavový diagram** bývá **velký** i pro jednoduché konkurentní programy – náročná konstrukce a hledání nežádoucích stavů u vzájemného vyloučení, cyklů vedoucích k uváznutí a vyhladovění atd.

= počítačový program pro **konstrukci stavového diagramu a ověřování korektnosti programu**

- ověřuje, zda stavový diagram je modelem formule v temporální logice specifikující vlastnosti korektnosti
- př. **Spin** – také simulátor konkurence

# API pro paralelní programování

- používající **(globální) sdílenou paměť**, ne posílání zpráv (= distribuované programování, např. PVM, MPI)
- realizace abstraktních paralelních procesů: procesy, **vlákna** (častěji) – spuštění nových z hlavního, po skončení „připojení“ (join)
- pro konkrétní prog. jazyk(y), platformu(y) a operační systém(y): C/C++/C#, POSIX, .NET

**Nízkoúrovňová (multiprocessing, multithreading)**

= **explicitní paralelizace** (správa procesů) **a synchronizace**, typicky **různé procesy**

- funkce/objekty pro **správu procesů** (vytvoření, ukončení), čekání na ukončení procesu
- dat. typy + funkce nebo objekty pro **synchronizační primitiva**: atomické akce (test-and-set, fetch-and-add atd.), semafory, kritické sekce, bariéry, monitory, podmíněné proměnné, ...

– větší kontrola za cenu potřeby znát paralelizační prostředí a „vědět, co dělám“

# POSIX Threads (Pthreads)

= POSIX standard pro práci s vlákny

- knihovna kromě unixových OS (standardní součást) i pro MS Windows (v rámci SFU/SUA, Pthreads-win32 nad Win32 API)
- C funkce pthread_*: správa vláken, mutexy, bariéra, podmíněná proměnná, aktivní (spin) a R/W zámek
- plus POSIX API pro semafory: C funkce sem_*

# Win32 Threading API (Windows Threads)

= součást Win32 API

- C funkce: správa vláken, kritická sekce, mutex, semafor, událost

# Boost C++ Libraries

= knihovny C++ šablon, podobné jako STL (text, kontejnery, iterátory, algoritmy, matematika, správa paměti aj.), knihovna thread (**Boost Threads**)

- třídy a funkce: správa vláken, mutexy, bariéra, podmíněná proměnná

# C++0x

= nový standard C++, t.č. (2011) ve fázi draftu, implementace Just Software

- kromě rozšíření prog. jazyka a std. knihovny další knihovny (z C++ TR1), podpora vláken
- třídy (jmenný prostor std): správa vláken, atomické operace, mutexy, podmíněné proměnné

# .NET vlákna

= součást .NET 2.0 a 3.5

- středněúrovňové – **fond vláken** (viz dále)
- třídy (jmenný prostor System.Threading): správa vláken Thread, BackgroundWorker a ThreadPool, atomické akce Interlocked.*, zámky, mutex, semafor, událost, monitor (atribut Synchronization), podmíněná proměnná

# API pro paralelní programování

**Vysokoúrovňová (tzv. „logická paralelizace“, task parallelism)**

= **implicitní paralelizace** (správa procesů) **a synchronizace**, typicky **stejné procesy**

– implicitní plánování procesů, tzv. **úloh (tasks)**, na vlákna z **fondu vláken (thread pool)**

- konstrukty prog. jazyka pro **paralelní vykonávání a synchronizaci úloh**: chráněné objekty a dat. struktury, paralelní cykly, algoritmy

– není nutné znát paralelizační prostředí za cenu menší kontroly

# Open Multi-Processing (OpenMP)

= součást překladače (GCC, MS Visual C++, Intel Compiler/Parallel Studio aj.)

- vyznačení částí C/C++/Fortran programu, které mají být vykonávány paralelně, pomocí direktiv preprocesoru jazyka (v C/C++) #pragma omp *
  - paralelní blok (i podmíněně), cyklus (nastavení statického/dynamického plánování), sdílené/privátní proměnné (sumarizace/redukce privátních do sdílené), atomická akce, kritická sekce, bariéra (vedle implicitní)
  - při sekvenčním překladu ignorovány
- funkce a konstanty: zjištění a nastavení počtu vláken, test prezence v paralelním kontextu vykonávání, zjištění počtu procesorů, zámky aj.
- i rozšíření na platformy bez sdílené paměti (Cluster OpenMP)

# Intel Threading Building Block (TBB)

= knihovna C++ šablon

- třídy a funkce: paralelní cykly, sumarizace/redukce a třídění parallel_*, atomické operace atomic<T>.*, mutex, dat. struktury (fronta, vektor, hashmap) concurrent_*, škálovatelná správa paměti scalable_*
- dynamické plánování vláken na jádra procesoru, umožněna i kontrola správy vláken (Task Scheduler)

# Parallel Extensions

= součást .NET 4.0

- dvě části: Parallel LINQ (PLINQ) a **Task Parallel Library (TPL)**, plus datové struktury pro synchronizaci úloh
- ukládání úloh do front pomocí Task Manageru a implicitní plánování jako vláken na jádra procesoru (pomocí fondu vláken)
- úloha = lambda funkce
- třídy (jmenný prostor System.Threading.Tasks): správa úloh Task, paralelní vykonávání úloh a cykly Parallel.* (implicitní vytvoření úloh a bariéra), kolekce Concurrent*

# Grand Central Dispatch (GCD)

= rozšíření prog. jazyka a knihovna pro Mac OS X, iOS, FreeBSD

- ukládání úloh do (dispatch) front nebo zdrojů (source) a implicitní plánování jako vláken na jádra procesoru (pomocí fondu vláken), po příp. (pro zdroj) vyvolání systémové události (časovač, síťový socket, souborový deksriptor aj.)
- úloha = funkce nebo tzv. **blok** = rozšíření C/C++/Objective-C zapouzdřující volání funkce a argumenty (podobné uzávěru)
- funkce/třídy: správa front (3 globální, privátní sériové) Dispatch Queues, zdrojů Dispatch Sources, skupin úloh Dispatch Groups (implicitní bariéra) a semaforů Dispatch Semaphores (jen několik úloh paralelně) dispatch_*

# Frameworky pro heterogenní platformy

- výpočetní zařízení CPU, GPU (pro obecné negrafické výpočty, **GPGPU**) aj.
- nízkoúrovňové, součástí prog. jazyk (založený na C) + překladač a API – C nebo tzv. language binding z jiného jazyka

= programování tzv. **jader (kernels)** – v jazyce napsané a přeložené funkce **paralelně vykonávány zařízeními** nad daty z tzv. **streamu** (vektory, matice, atd.), řízení (překlad a spuštění funkcí, výměna dat, pomocí API) programem na hostitelském zařízení (CPU)

- synchronizace: bariéra, události (podmíněné proměnné)
- použití na výpočty všeho druhu
- **OpenCL**, ATI Stream, Nvidia CUDA, MS DirectCompute
- implementace v ovladačích grafických karet, matematických SW