

Vláknové programování

část I

Lukáš Hejmánek, Petr Holub
{`xhejtman, hopet`}@ics.muni.cz



Laboratoř pokročilých síťových technologií

PV192
2014-02-18

“For the past thirty years, computer performance has been driven by Moore’s Law; from now in, it will be driven by Amdahl’s Law. Writing code that effectively exploits multiple processors can be very challenging. . .”

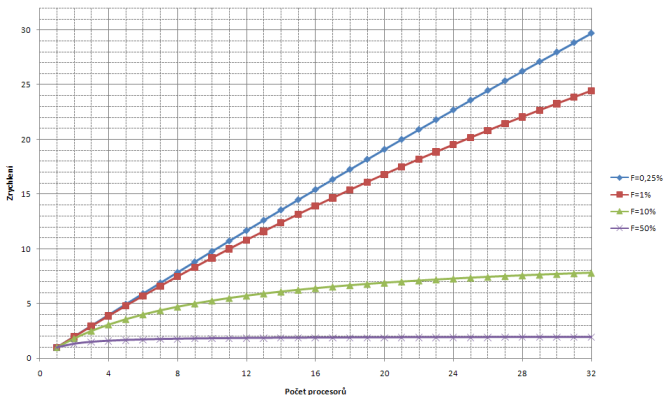
–*Doron Rajwan*, Research Scientist, Intel Corp.

Amdahlův zákon

$$\text{zrychlení} \leq \frac{1}{F + \frac{1-F}{N}}$$

kde F je podíl sériově vykonávané práce a N je počet procesorů

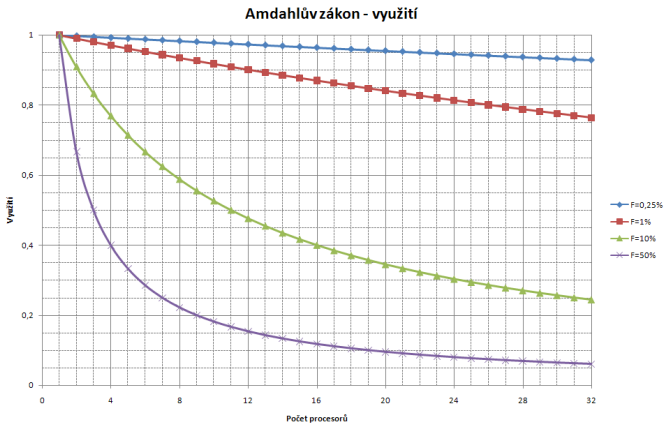
Amdahlův zákon - zrychlení



Amdahlův zákon

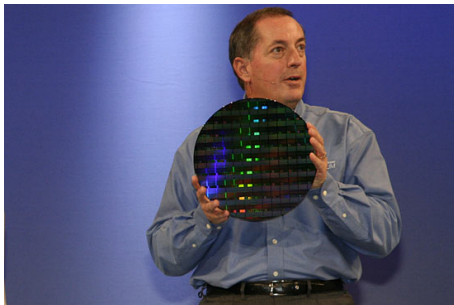
$$\text{zrychlení} \leq \frac{1}{F + \frac{1-F}{N}}$$

kde F je podíl sériově vykonávané práce a N je počet procesorů



80 jader na čipu

Intel na IDF 2006 předvedl prototyp procesoru s 80 jádry.
Výkon \approx 1 TFLOPS



Zdroje:

http://www.news.com/Intel-shows-off-80-core-processor/2100-1006_3-6158181.html

Úvod do programování ve vláknech

- Programování v C s využitím Pthreads
- Programování v Javě
- Výlet do jiných jazyků: Ada
- Demonstrováno na praktických příkladech

Úvod do programování ve vláknech

- Programování v C s využitím Pthreads
 1. Procesy, vlákna, přepínání kontextu, knihovna pthreads, vznik a ukončení vláken, základy ladění aplikací
 2. Základy synchronizace: zámky, semaforey, podmíněné proměnné
 3. Pokročilé synchronizace: bariéry, rw zámky, pojmenované semaforey, futexy
 4. Afinita, Atributy vláken, režimy startu vlákna, priority, ukončování vláken, thread-specific data
 5. OpenMP
 6. Práce s pamětí
 7. GUI, OpenGL, Futures a TPE v C++

Úvod do programování ve vláknech

- Programování v Javě
 1. Vlákna v jazyce Java, vytváření a ukončování. Viditelnost operací. Monitory a synchronizace. Signalizace a pozastavení.
 2. Paralelní datové struktury. Atomické typy. Ladění paralelních programů: uváznutí a jeho diagnostika, hladovění. Testování paralelních programů.
 3. Explicitní zamykání – RW zámky, vlastní typy zámků. Executors, thread pools, futures.
 4. Paměťový model Javy. Paralelismus a GUI.

Úvod do programování ve vláknech

- Výlet do jiných jazyků: Ada
 1. Úvod k jazyku Ada: základní rysy jazyka, syntaxe, datové typy.
 2. Podpora paralelismu v jazyce Ada: úlohy, chráněné objekty, monitory, podpora systémů v reálném čase.

Literatura

- Andrews, Gregory R. *Foundations of multithreaded, parallel, and distributed programming*. Addison-Wesley 2000.
- Ben-Ari M., *Principles of Concurrent and Distributed Programming*. 2nd Ed. Addison-Wesley, 2006
- Goetz B., Peierls T., Bloch J., Bowbeer J., Holmes D., Lea D. *Java Concurrency in Practice*. Addison Wesley Professional, 2006
- Butenhof D. R., *Programming with POSIX(R) Threads*. Addison-Wesley Professional, 1997
- Burns A., Wellings A. *Concurrency in Ada*. 2nd Ed. Cambridge University Press, 1998
nebo
Burns A., Wellings A. *Concurrent and Real-Time Programming in Ada*. Cambridge University Press, 2007

Přehled přednášky

Procesy a vlákna

Procesy

Vlákna

Procesy

- Proces
 - Instance programu, která je sekvenčně prováděna.
 - Je to entita pro alokace zdrojů (procesor, paměť, atd)
 - Procesy tvoří stromovou hierarchii—vztah rodič potomek

Sex is not really common among processes—each process has just one parent.

Procesy

- Typy procesů
 - Levný proces (Light Weight Process–LWP)
 - Levné procesy mezi sebou sdílí adresní prostor
 - Minimum privátních zdrojů
 - Drahý proces (Heavy Weight Process–HWP)
 - Drahé procesy jsou mezi sebou zcela izolované
 - Prakticky všechny zdroje jsou privátní

Procesy

- Popisovač procesu
 - Obsahuje informace o
 - Signálech
 - Přidělené paměti
 - Otevřených souborech
 - Aktuálním adresáři
 - HW kontext (obsah registrů, zásobník, ...) – TSS
 - Terminálu
 - Prioritě
 - Stav
 - ...

Procesy

- Vytvoření procesu
 - Proces vzniká rozštěpením rodiče
 - Po startu je potomek stejný jako rodič
 - Stejný obsah paměti (Copy on Write)
 - Vykonává stejný kód

Procesy

- Běh procesu
 - Vykonávání kódu programu – charakterizován kontextem
 - Dva režimy běhu
 - User space–kód samotného programu
 - Kernel space–kód jádra
 - Stav procesu
 - Running
 - Interruptible
 - Uninterruptible
(Nezpracovává signály)
 - Stopped
 - Traced
 - Konkurence vs. paralelismus
 - Konkurence–vykonávání stejného nebo různého kódu více procesy, nemusí probíhat ve stejný čas
 - Paralelismus–konkurence probíhající ve stejný čas

Procesy

- Přepínání kontextu
 - Zásadní mechanismus multitaskingu
 - Mechanismus uložení a obnovení stavu CPU
 - Rozlišujeme přepnutí kontextu
 - Registrové (obsluha přerušení)
 - Vlákňové (přepnutí na jiné vlákno téhož procesu)
 - Procesové (přepnutí na jiný proces)

Procesy

- Kroky při přepínání kontextu
 - Uložení stavu CPU, obvykle do TSS
 - Všechny běžné registry, deskriptory segmentu, příznaky
 - Stav a registry FPU
 - Obnova adresního prostoru
 - Načtení nového stavu CPU

Procesy

- Softwarové vs. hardwarové přepínání kontextu
 - Kontext lze uložit a obnovit v softwaru (kopírování stavu)
 - Některé procesory podporují přepnutí kontextu v HW (architektura x86 od Intel 80386 a dál)
 - Linux od verze jádra 2.4 používá softwarové přepnutí kontextu
 - Softwarové i hardwarové přepnutí kontextu je velmi drahá operace!

Procesy

- Komunikace mezi procesy
 - Soubory
 - Signály (`signal(7)`)
 - Sockets (`socket(2)`)
 - Fronty zpráv (`mq_overview(7)`)
 - Trubky (pipes), pojmenované vs. nepojmenované (`pipe(2)`)
 - Semaforey (`sem_overview(7)`)
 - Sdílená paměť, paměťově mapované soubory (`shm_overview(7)`, `mmap(2)`)
 - Message passing (MPI knihovny)

Vlákna

- Proč vlákna?
 - Paralelismus
 - Výkon
 - Odezva
 - Komunikace

Vlákna

- Vlákno
 - Podmnožina procesu
 - Vlákno nemá vlastní adresní prostor
 - Typicky sdílí stav ostatními vlákny daného procesu
 - Shared-memory model:
 - Komunikace mezi vlákny je možná stejně jako u procesů a navíc i přes jejich sdílenou paměť
 - Oproti procesu jsou běžně sdílené globální a statické proměnné

Vlákna

- Vlákna stejného procesu sdílí
 - Kód programu
 - „Většinu“ dat
 - Novější koncepce vláken podporuje nesdílenou paměť–thread local storage (TLS)
 - Otevřené soubory (file descriptors)
 - Signály a obsluhu signálů
 - Současný pracovní adresář
 - Identifikaci uživatele a skupiny
 - Process ID (PID)
 - Pojmenované semaforey, fronty zpráv a další nástroje IPC

Vlákna

- Každé vlákno má unikátní
 - Thread ID (identifikace vlákna)
 - Obsah registrů procesoru, ukazatel vrcholu zásobníku
 - Zásobník pro lokální proměnné a návratové adresy
 - Masku signálů
 - Prioritu
 - Hodnotu proměnné **errno** (dle POSIX.1c)

Vlákna

- Implementace vláken v operačním systému je různá
 - proces a vlákno není rozlišeno
(vlákno je tedy procesem – Linux bez NPTL)
 - proces a vlákno jsou rozlišeny
(vlákno se liší od procesu – Windows, Linux s NPTL)
 - vlákna v uživatelském prostoru
(vlákna si řídí sám proces – Java Green Threads (obsolete), Erlang)

Vlákna

- Mapování vláken na plánovací entity v jádře
 - mapování vláken 1:1
 - současné produkční implementace
 - Linux, Windows, FreeBSD s libthr
 - mapování vláken N:M
 - + teoreticky nejefektivnější
 - příliš složité, problémy s invertováním priorit, atd.
 - FreeBSD s Kernel Scheduler Entities, experimenty i v Linuxu
 - mapování vláken N:1
 - zastaralý přístup, user-space threading
 - FreeBSD s libc_r

Vlákna

- Některé systémy mají „levné“ přepínání vláken a „drahé“ přepínání procesů (Windows NT, OS/2).
- Některé systémy příliš nerozlišují v přepnutí vlákna nebo procesu.
- Proč uvažujeme vlákna místo procesů?
 - Rychlejší přepnutí běžících vláken než běžících procesů.
 - Snadné sdílení paměti a dalších zdrojů mezi běžícími vlákny (někdy ovšem nevýhoda).