

IA039

Měření času, profilování a měření výkonu

Měření času výpočtu

- Optimalizace není možná bez znalosti, co optimalizovat
- Potřebujeme znát údaje o běhu programu
 - Čas výpočtu celého programu: příkazy **time** a **timex**
 - Čas výpočtu částí programů: profilování
 - Srovnání systémů: benchmarking

Příkaz time

- User time
 - Čas procesoru strávený uživatelskými procesy
- System time
 - Čas procesoru strávený obsluhou funkcí jádra
- Elapsed time
 - Celkový čas výpočtu

CPU time = user time + system time

Příkaz time – další údaje

- Dostupné pro **time** v prostředí C shellu
- Sdílený paměťový prostor
- Privátní (unshared) paměťový prostor
- Počet block input operací
- Počet block output operací
- Počet page faultů
- Počet swapů

Příkaz `timex` (System V)

- Údaje o čase
 - real, user a system time
- Další údaje
 - Mean memory size
 - Kcore-minutes
 - Přenesené bloky a znaky
- Využívá `sar` accounting

Příklady – time

```
time netscape http://www.pgroup.com/ppro_cc_desc.html  
7.3u 3.1s 1:31 11% 0+0k 272+36io 125pf+0w
```

Příklady – timex

```
timex -opkmt netscape http://www.pgroup.com/ppro_cc_desc.html
```

```
real      2:33.61
```

```
user       9.20
```

```
sys        3.50
```

```
START AFT: Mon May  4 10:21:19 1998
```

```
END BEFOR: Mon May  4 10:23:53 1998
```

```
COMMAND           START      END      REAL
```

```
NAME      USER  TTYNAME  TIME      TIME      (SECS)
```

```
netstat   ludek  pts/5   10:21:26 10:21:27   1.26
```

```
netscape  ludek  pts/5   10:21:19 10:23:52 153.60
```

```
CPU (SECS)  CHARS  BLOCKS  MEAN  KCORE
```

```
SYS  USER  TRNSFD  READ  SIZE(K)  MIN
```

```
0.09  0.85  577536  32    104.34  1.63
```

```
3.40  8.34  7045120  486   1144.37  223.91
```

```
CHARS TRNSFD = 7622656
```

```
BLOCKS READ  = 518
```

Profiling

- Snaha získat informace o částech programu
 - Doba strávená v jednotlivých blocích
 - Doba strávená jednotlivými příkazy
 - Počet opakování bloků/příkazů
- Primární pozornost zaměřena na procedury
- Profil: graf
 - Osa X: jednotlivé procedury
 - Osa Y: Doba výpočtu

Typy profilů

■ Ostrý profil

- Píky odpovídají dominujícím procedurám
- Numerické aplikace, technické výpočty s maticemi atd.
- „Snadno“ optimalizovatelné

■ Plochý profil

- Program tráví čas rovnoměrně ve všech procedurách
- Zpravidla databáze, informační systémy, systémové programy
- Obtížně optimalizovatelné

■ Amdahlovo pravidlo platí i zde

Profilery

- Nástroje pro tvorbu profilů
 - Samozřejmě možné i „ručně“
- Procedurově orientované
 - **prof**
 - **gprof**
- Blokově (řádkově) orientované
 - **pixie**
 - **tcov**
 - **lprof**

Použití profilerů

- Dvě fáze:
 - Instrumentovaný běh programu (s nebo bez opětného překladu)
 - Vlastní zpráva o výsledcích (report)
- Přístup ke zdrojovému kódu
 - Znalost struktury programu

Procedurově orientované profily

- Typický představitel: **prof**
- Instrumentace
 - Nový překlad programu
 - Zpravidla dostupný přes přepínač překladače **-p**
- Výpočet
 - Instrumentovaný program vytváří záznam o výpočtu
 - Soubor **mon.out**
- Vytvoření zprávy
 - Vlastní běh programu **prof**

prof – další informace

- Okročilé procesory a operační systémy nepotřebují zvláštní překlad
 - Např. SGI IRIX 6.x (procesory MIPS R1X000)
 - Instrumentace za běhu – SpeedShop
 - * Program spouštěn příkazem **ssrun -exptype**
 - * Výpočet generuje soubor `<a.out-name>.<exptype>.<code><pid>`
 - * **exptype** definuje typ profilu

SpeedShop – typy profilů

- `usertime` – uživatelský čas
- `[f]pcsamp[x]` – vzorkování
- `ideal (pixie)` – blokový profiler
- `fpe` – pohyblivá řádková čárka
- `prof_hwc` – hardwarové čítače

`f gi_hwc, [f]cy_hwc, [f]ic_hwc, [f]isc_hwc, [f]dc_hwc, [f]dsc_hwc, [f]tlb_hwc,
[f]gfp_hwc`

gprof

- Poskytuje call graf
 - Kdo koho volá
- Počty volání procedur
- Kumulované časy výpočtu větví

Přesnost měření

- Měření času

- Absolutní čas vstupu a výstupu procedury

- * Vnořené procedury

- * Krátké procedury

- Sběr hodnoty čítače instrukcí v pravidelných intervalech

- * Přesnost výsledku závislá na vzorkování (sampling interval)

Blokově orientované profily

- Poskytují informace o průchodech základními bloky
- **tcov a lprof**
 - Počet průchodů každým příkazem (řádkem)
- **pixie**
 - Počet cyklů procesorů strávený v každém příkazu

Příklad

```
1: static void foo(), bar(), baz();
2:
3: main()
4: { int l;
5:   for (l=0; l <1000; l++)
6:     { if ( l == 2*(l/2) )
7:       foo();
8:       bar();
9:       baz();
10:    }
11: }
12:
13: void foo()
14: { int j;
15:   for (j=0; j<200; j++);
16: }
17: void bar()
18: { int j;
19:   for (j=0; j<200; j++);
20: }
21: void baz()
22: { int j;
23:   for (j=0; j<300; j++);
24: }
```

SpeedShop – usertime

usertime:

index	%Samples	self	descendents	total	name
[1]	100.0%	0.00	0.03	1	main
[2]	100.0%	0.03	0.00	1	bar

SpeedShop – vzorkování

pcsamp:

samples	time(%)	cum time(%)	procedure
2	0.02s(50.0)	0.02s(50.0)	foo
1	0.01s(25.0)	0.03s(75.0)	bar
1	0.01s(25.0)	0.04s(100.0)	baz

fpcsamp:

samples	time(%)	cum time(%)	procedure
18	0.02s(41.9)	0.02s(41.9)	baz
12	0.01s(27.9)	0.03s(69.8)	bar
12	0.01s(27.9)	0.04s(97.7)	foo

SpeedShop – blokový přístup

ideal:

cycles(%)	cum %	secs	instrns	alls	procedure
3918000(49.63)	49.63	0.03	2111000	1000	baz
2618000(33.16)	82.80	0.02	1411000	1000	bar
1309000(16.58)	99.38	0.01	705500	500	foo
47024(0.60)	99.98	0.00	25017	1	main

SpeedShop – blokový přístup II

ideal -h:

	cycles(%)	cum %	times	line	procedure
	3907858(49.50%)	49.50%	300000	23	baz
	2607858(33.04%)	82.54%	200000	19	bar
	1303930(16.52%)	99.06%	100000	15	foo
	14000(0.18%)	99.24%	1000	6	main
	13009(0.16%)	99.40%	1000	5	main
	8000(0.10%)	99.50%	1000	9	main
	8000(0.10%)	99.60%	1000	8	main
	7000(0.09%)	99.69%	1000	24	baz
	7000(0.09%)	99.78%	1000	20	bar
	4000(0.05%)	99.83%	500	7	main
	3500(0.04%)	99.88%	500	16	foo
	3142(0.04%)	99.92%	1000	18	bar
	3142(0.04%)	99.96%	1000	22	baz
	1570(0.02%)	99.98%	500	14	foo

tcov

tcov:

```
1 -> for (l=0; l <1000; l++)
1000 -> { if ( l == 2*(l/2) )
500 ->     foo();
1000 ->     bar();
        baz();
```

```
void foo()
```

```
500 -> for (j=0; j<200; j++);
```

```
void bar()
```

```
1000 -> for (j=0; j<200; j++);
```

```
void baz()
```

```
1000 -> for (j=0; j<300; j++);
```

tcov – pokračování

Top 10 Blocks

Line	Count
6	1000
8	1000
19	1000
23	1000
7	500
15	500
5	1

7 Basic blocks in this file

7 Basic blocks executed

100.00 Percent of the file executed

5001 Total basic block executions

714.43 Average executions per basic block

Zjišťování výkonu – benchmarking

- Snaha o porovnání *systemů*
 - Hardware i software společně
- Neexistuje žádné „zázračné“ řešení
- Základní přístupy
 - Průmyslové („profesionální“) benchmarky
 - * Porovnatelnost, nezávislost na výrobcích
 - „Privátní“ benchmarky
 - * Konkrétní (specifické) požadavky

Mysteriózní MIPS a MFLOPS

- Srovnání na základě počtu instrukcí vykonaných za sekundu
- MIPS – milion celočíselných instrukcí za sekundu
- MFLOPS – milion operací s pohyblivou řádovou čárkou za sekundu
- Problémy
 - Jaké instrukce?
 - V jaké posloupnosti?
- Umělé, nevypovídající

Celočíselné benchmarky

- VAX MIPS
- Dhrystones

Benchmarky s pohyblivou řádovou čárkou

- Whetstone (umělý mix, skalární)
- Linpack (daxpy, vektorizace)
 - 100*100
 - 1000*1000

SPEC benchmarks

- Nezávislá organizace
 - Standard Performance Evaluation Corporation
- Standardizované benchmarky pro různé architektury
- Vychází z tzv. *kernel kódů*
 - Celý nebo část existujícího programu
 - Dostupné ve zdrojovém kódu
 - * Je možno „doladit“

SPEC skupiny

- Open Systems Group (OSG)
- High Performance Group (HPG)
- Graphics Performance Characterization Group (GPC)

SPEC OSG podskupiny

- CPU

- SPECmarks a CPU benchmarks

- JAVA

- JVM98, JBB2005, Java client a server benchmarky

- MAIL

- SPECmail2001

- SFS

- Systémy souborů (SFS97)

- WEB

- WEB99, WEB99_SSL, WEB2005

CPU2000

- Aktuální CPU benchmark
- Dělení
 - CINT2000 – celočíselné výpočty
 - CFP2000 – výpočty s pohyblivou řádovou čárkou

CINT2000

■ Jednotlivé součásti

- 164.gzip Compression
- 175.vpr FPGA Circuit Placement and Routing
- 176.gcc C Programming Language Compiler
- 181.mcf Combinatorial Optimization
- 186.crafty Game Playing: Chess
- 197.parser Word Processing
- 252.eon Computer Visualization
- 253.perlbnk PERL Programming Language
- 254.gap Group Theory, Interpreter
- 255.vortex Object-oriented Database
- 256.bzip2 Compression
- 300.twolf Place and Route Simulator

CFP2000

■ Jednotlivé součásti

- 168.wupwise Physics / Quantum Chromodynamics
- 171.swim Shallow Water Modeling
- 172.mgrid Multi-grid Solver: 3D Potential Field
- 173.applu Parabolic / Elliptic Partial Differential Equations
- 177.mesa 3-D Graphics Library
- 178.galgel Computational Fluid Dynamics
- 179.art Image Recognition / Neural Networks
- 183.quake Seismic Wave Propagation Simulation
- 187.facerec Image Processing: Face Recognition
- 188.amp Computational Chemistry
- 189.lucas Number Theory / Primality Testing
- 191.fma3d Finite-element Crash Simulation
- 200.sixtrack High Energy Nuclear Physics Accelerator Design
- 301.apsi Meteorology: Pollutant Distribution

Transakční benchmarky

- Výkon databází
 - TPC-A
 - * Testuje interakci s ATM (6 požadavků za minutu)
 - * 1 TPS znamená, že 10 ATM současně vydá požadavek a výsledek dostane do 2 s (s 90 % účinností)
 - TPC-B
 - * Jako TPC-A, ale testuje se přímo, ne přes (pomalou) síť
 - TPC-C
 - * Komplexní, transakce jsou objednávky, platby, dotazy s jistým procentem záměrných chyb vyžadujících automatickou korekci

Síťové benchmarky

- netperf
- iperf
- End to end měření

Vlastní benchmarky

- Specifické (konkrétní) požadavky
- Důležité parametry:
 - Co testovat
 - Jak dlouho testovat
 - Požadavky na velikost paměti
- Typy benchmarků
 - Jednoduchý proud (opakování)
 - Propustnost (benchmark stone wall)

Kontroly

- Nezbytná součást používání benchmarků
 - Měříme opravdu to, co chceme?
- Možné příčiny obvlivnění
 - Použitá optimalizace
 - Velikost paměti
 - Přítomnost jiných procesů
- Co je třeba explicitně kontrolovat
 - CPU čas a čas nástěných hodin
 - Výsledky!
 - Srovnání se „známým“ standardem