Architektura počítačových systémů

Róbert Lórencz

1. přednáška

Kvantitativní principy návrhu počítačů

http://service.felk.cvut.cz/courses/36APS
lorencz@fel.cvut.cz

Obsah přednášky

- historie
- časová složitost, propustnost, výkonnost
- Amdahlův zákon
- CPU výkonnostní rovnice
- přesnější CPU výkonnostní rovnice
- zkušební úlohy
- MIPS a MFLOPS

Historie – vývojové mezníky

- víc než 50 let uplynulo od vytvoření 1. univerzálního elektronického počítače
- dnešní PC za 1000 $ jsou výkonnější než počítač z r. 1980 za 1 mil. $
- HW průlom: VLSI technologie a příchod mikroprocesorů (70. léta)
- SW průlom: univerzální na výrobci nezávislé OS (UNIX) a přechod od programování v SOJ k programování ve vyšších jazycích
- nástup RISC (Reduced Instruction Set Computer), důsledek:
  - paralelizmus na úrovni zpracování instrukcí – ILP (Instruction Level Parallelism), tj. proudové zpracování instrukcí, super-skalární architektury atd.
  - používání vnitřních skrytých pamětí (cache)
- průlom v navrhování: vývoj kvantitativního přístupu k návrhu a analýze počítačů, který využívá empirická pozorování, experimentování a simulace

Historie - chronologie

- 60. léta: dominantní velké sálové počítače s aplikacemi jako:
  - zpracování dat ve finanční sféře
  - rozsáhlé vědeckotechnické výpočty
- 70. léta: mikropočítače pro aplikace ve vědeckých laboratořích
- 80. léta: příchod stolních počítačů založených na mikroprocesorech (osobní počítače a pracovní stanice)
- dále se objevují servery a lokální sítě pro větší úlohy s větší pamětí a výkonem
- 90. léta: Internet a WWW technologie
- současnost: rozdělení počítačového trhu na 3 oblasti charakterizované rozdílným použitím, požadavky a počítačovou technologií:
  1. osobní, stolní a přenosné počítače
  2. servery a výkonné paralelní počítače a superpočítače
  3. vestavěné a řídící počítače v jednoúčelových zařízeních

Časová složitost, propustnost, výkonnost

Letadlo	Doba letu t DC - Paříž [h]	Rychlost v [km/h]	Kapacita c [osoba]	Propustnost $r = c \cdot v$ [osoba $\times$ km/h]
Boeing 747	6.5	981	470	461070
Concorde	3	2172	132	286704

Která linka má větší propustnost / výkonnost?

- doba letu (doba výpočtu, čas odezvy, latence) = čas, za který splní daný úkon
- propustnost (šířka pásma, výkon) = práce za den, hodinu, týden, s, . . .

Počítačové analogie:

- # vykonaných instrukcí v procesoru
- # přenesených paketů v síti
- propustnost dat na sběrnici atd.

Časová složitost a výkonnost

Concorde vs. Boeing 747 $\rightarrow$ doba letu různá, ale úkon stejný
Úkon: let DC – Paříž $\rightarrow$ nezáleží na počtu přepravených osob
Doba letu: Concorde T_C = 3 h, Boeing T_B = 6.5 h
Concorde je $T_B/T_C = 6.5/3 = 2.2$ krát rychlejší

Výkonnost $P(T)$: inverzní hodnota doby T provedení 1 úkonu

$$P_T(T) = \frac{1}{\text{časová složitost 1 úkonu}} = \frac{1}{\text{let DC - Paříž}} = \frac{1}{T}$$

Počítačová analogie:

- úkon: provedení 1 programu
- doba: doba T provedení programu

X je k krát rychlejší než Y
X má k krát větší výkonnost než Y

$$\Leftarrow \quad k = \frac{P_T(T_x)}{P_T(T_y)} = \frac{T_y}{T_x}$$

Časová složitost a propustnost

Concorde vs. Boeing 747: doba letu různá, ale také různě velký úkon
Úkon: # přepravených osob DC – Paříž
Concorde c = 132 os., Boeing c = 470 os.
Doba letu: Concorde T_C = 3 h, Boeing T_B = 6.5 h
Boeing je $r_B/r_C = 461070/286704 = 1.6$ krát propustnější

Výkonnost $P(n,T)$: n úkonů za čas T

$$P_r(n,T) = \frac{\text{úkon}}{\text{čas. složitost}} = \frac{\#\text{ osob} \times \text{vzdál.}}{\text{let DC - Paříž}} = \frac{n}{T} = \frac{cd}{T} = cv = r$$

Počítačová analogie:

- úkon: různý, provedení n krát jednoho programu
- doba: doba T pro provedení úkonu

X je k krát propustnější než Y
X má k krát větší výkonnost než Y

$$\Leftarrow \quad k = \frac{P_r(n_X,T_X)}{P_r(n_Y,T_Y)} = \frac{T_Y n_X}{T_X n_Y}$$

Amdahlův zákon 1

Amdahlův zákon (AZ) = výpočet výkonového zisku, čili zrychlení S, dosaženého vylepšením nějaké části počítače

$$S = \frac{\text{výkonnost při využití vylepšení}}{\text{výkonnost bez využití vylepšení}} = \frac{P_{NEW}}{P_{OLD}}$$

nebo

$$S = \frac{\text{doba výpočtu bez využití vylepšení}}{\text{doba výpočtu při využití vylepšení}} = \frac{T_{OLD}}{T_{NEW}}$$

Zrychlení S = číslo, kolikrát je rychlejší běh úlohy na počítači s vylepšením oproti běhu stejné úlohy na původním počítači

Amdahlův zákon 2

Definujeme:

$$F_E = \frac{\text{původní doba výpočtu zlepšené části úlohy}}{\text{původní celková doba výpočtu}} \leq 1$$

nebo

$$S_E = \frac{\text{původní doba výpočtu zlepšené části úlohy}}{\text{doba výpočtu zlepšené části úlohy}} > 1$$

Doba výpočtu T_{NEW} na vylepšeném počítači se bude skládat z:

$(1 - F_E)T_{OLD}$ = doba výpočtu té části úlohy, kterou nelze vylepšit

$\frac{F_E}{S_E}T_{OLD}$ = doba výpočtu vylepšené části úlohy

Amdahlův zákon 3

Doba výpočtu T_{NEW} na vylepšeném počítači:

$$T_{NEW} = T_{OLD} \left((1 - F_E) + \frac{F_E}{S_E} \right)$$

Pro celkové zrychlení $S_{OVERALL}$ odpovídající danému vylepšení:

$$S_{OVERALL} = \frac{T_{OLD}}{T_{NEW}} = \frac{1}{(1 - F_E) + \frac{F_E}{S_E}}$$

Amdahlův zákon 4

Příklad 1

Předpokládejme, že chceme vylepšit procesor serveru pro Web. Nový CPU je desetkrát rychlejší pro Web aplikace než nynější. Dále víme, že nyní je CPU zaměstnáno ze 40% výpočty a 60% času je čekání na I/O.

Jaké bude celkové zrychlení po plánovaném vylepšení?

Řešení:

$$F_E = 0.4, \quad S_E = 10$$

$$S_{OVERALL} = \frac{1}{(1 - 0.4) + \frac{0.4}{10}} = \frac{1}{0.64} = 1.56$$

Amdahlův zákon 5

Příklad 2

Pro FP výpočty v počítačové grafice je hodně používaná operace odmocniny FPSQRT a výkonnost procesorů pro grafiku je na jejím efektivním provádění silně závislá.
Předpokládejme, že FPSQRT odpovídá 20% a všechny FP instrukce odpovídají 50% doby výpočtu kritické zkušební úlohy pro grafiku.
Úkolem je rozhodnout, zda je výhodnější:

1. 10 x zrychlit provádění instrukce FPSQRT, nebo
2. 1.6 x zrychlit provádění všech FP instrukcí?

Řešení:

1. $S_{FPSQRT} = \frac{1}{(1-0.2)+\frac{0.2}{10}} = \frac{1}{0.82} = 1.22$
2. $S_{FP} = \frac{1}{(1-0.5)+\frac{0.5}{1.6}} = \frac{1}{0.8125} = 1.23 \leftarrow$ výhodnější řešení

CPU výkonnostní rovnice 1

Výkonnost CPU vyjádřená pomocí doby $T_{CPU}(prg)$ pro vykonání programu *prg* jednotkou CPU je

$$P(T_{CPU}(prg)) = \frac{1}{T_{CPU}(prg)}$$

Všechny počítače používají hodiny s konstantními frekvencemi = cykly (ticks, clock ticks, clock periods, cycles, clock cycles). Rychlost hodin se udává:

- dobou trvání hodinového cyklu (clock period) - T_{CLK}, nebo
- frekvencí taktů (clock rate) - f_{CLK}

$$f_{CLK} = \frac{1}{T_{CLK}}$$

CPU výkonnostní rovnice 2

Doba CPU pro vykonání daného programu *prg* je

CPU výkonnostní rovnice

$$T_{CPU}(prg) = T_{CLK} \times Cyc_{CPU}(prg) = (1/f_{CLK}) \times Cyc_{CPU}(prg)$$

$Cyc_{CPU}(prg)$ = # hodinových cyklů CPU pro provedení programu *prg*.

Pokud známe počet instrukcí *IC* (instruction count) pro provedení programu *prg*, můžeme vypočítat *CPI* = průměrný počet hodinových cyklů na instrukci (clock cycles per instruction), nebo hodnotu *IPC* (instructions per clock cycle).

$$CPI(prg) = \frac{1}{IPC(prg)} = \frac{Cyc_{CPU}(prg)}{IC(prg)}$$

CPU výkonnostní rovnice 3

Doba CPU pro vykonání daného programu *prg* s proměnnými *IC* a *CPI* je

CPU výkonnostní rovnice

$$T_{CPU}(prg) = IC(prg) \times CPI(prg) \times T_{CLK} = IC(prg) \times CPI(prg) \times (1/f_{CLK})$$

Po převedení komponentů této rovnice do měřicích jednotek dostáváme

$$\left[\frac{\text{instrukce}}{\text{program}} \times \frac{\text{hod. cyklus}}{\text{instrukce}} \times \frac{\text{sekunda}}{\text{hod. cyklus}}\right] = \left[\frac{\text{sekunda}}{\text{program}}\right] = T_{CPU}(prg)$$

Rozklad $T_{CPU}(prg)$ demonstruje závislost $T_{CPU}(prg)$ na 3 parametrech:

1. hodinový cyklus T_{CLK} – hodinová frekvence f_{CLK}
2. *CPI* = # hodinových cyklů na jednu instrukci daného programu *prg*
3. *IC* = # instrukcí daného programu *prg*

Závislost parametrů T_{CPU}

Pokud se některý z parametrů zlepší o k %, pak se zlepší o k % i T_{CPU}. Parametry T_{CLK}, CPI, IC jsou provázány:

	IC	CPI	T_{CLK}
Program	●	●	
Překladač	●	●	
Architektura instrukčního souboru (ISA)	●	●	●
Organizace CPU		●	●
Technologie CPU			●

- ● – přímý vliv
- ● – není součástí systému
- ● – nepřímý, zprostředkovaný vliv

Přesnější CPU výkonnostní rovnice 1

Přesnější metrika:

$$Cyc_{CPU}(prg) = \sum_{i=1}^{n} (ic_i \times cpi_i)$$

$$T_{CPU}(prg) = \left(\sum_{i=1}^{n} ic_i \times cpi_i \right) \times T_{CLK}$$

kde

ic_i = # provedení instrukcí i programu prg,

cpi_i = průměrný # hodinových cyklů instrukce i,

n = # instrukcí v architektuře instrukčního souboru (ISA - Instruction Set Architecture).

Přesnější CPU výkonnostní rovnice 2

Průměrné *CPI*

$$CPI(prg) = \frac{\sum_{i=1}^{n} (ic_i \times cpi_i)}{IC(prg)} = \sum_{i=1}^{n} \left(\frac{ic_i}{IC(prg)} \times cpi_i \right)$$

- cpi_i zahrnuje vliv čekacích stavů, výpadků vnitřní skryté paměti atd.
- cpi_i lze měřit a počítat (v některých případech)

Dále je uveden příklad, kde průměrná hodnota *CPI* je vypočítána na základě znalosti *IC* a hodnot:

- ic_i = # instrukcí typu ALU, Load, Store a Branch
- cpi_i průměrná hodnota pro instrukce typu ALU, Load, Store a Branch

Přesnější CPU výkonnostní rovnice 3

Příklad typického mixu instrukcí RISC procesoru:

Instrukce	ic_i/IC	cpi_i	$(ic_i/IC) \times cpi_i$	% čas
ALU	0.5	1	0.5	23 %
Load	0.2	5	1	45 %
Store	0.1	3	0.3	14 %
Branch	0.2	2	0.4	18 %
CPI			2.2	

Otázky:

1. Jakým způsobem se urychlí celkový výpočet, pokud vylepšením datové vnitřní skryté paměti klesne cpi_{Load} na 2?
2. Předchozí zlepšení porovnejte s použitím vylepšení predikce skoku, které zmenší cpi_{Branch} na 1 ?
3. Co se stane, když budou 2 ALU instrukce vykonávány najednou?

Přesnější CPU výkonnostní rovnice 4

Příklad 3

Z měření instrukčního mixu jsme získali následující hodnoty:

- 25% provedených instrukcí jsou FP operace
- *cpi* FP instrukcí = 4.0
- průměrná hodnota *cpi* ostatních operací = 1.33
- FPSQRT instrukcí činí 2% provedených instrukcí
- *cpi* FPSQRT instrukce = 20

Úkolem je rozhodnout, zda je výhodnější:

1. zmenšit *cpi* FPSQRT na 2, nebo
2. zmenšit průměrnou hodnotu *cpi* všech FP instrukcí na 2.5?

Řešení:

f_{CLK} a IC zůstávají stejné. Pro původní CPI platí:

$$CPI_{ORIGINAL} = \sum_{i=1}^{n} \left(\frac{ic_i}{IC} \times cpi_i \right) = (0.25 \times 4) + (0.75 \times 1.33) = 2.0$$

Přesnější CPU výkonnostní rovnice 5

1. Pro *CPI* vylepšeného počítače u FPSQRT platí:
$$CPI_{NEW\_FPSQRT} = CPI_{ORIGINAL} - 0.02 \times (cpi_{OLD\_FPSQRT} - cpi_{NEW\_FPSQRT}) = 2.0 - 0.02 \times (20 - 2) = 1.64$$
2. Pro *CPI* počítače vylepšeného u všech FP platí:
$$CPI_{NEW\_FP} = (0.25 \times 2.5) + (0.75 \times 1.33) = 1.623$$

Zrychlení počítače vylepšeného podle 2. varianty oproti původnímu je

$$S_{NEW\_FP} = \frac{T_{CPU-ORIGINAL}}{T_{CPU-NEW\_FP}} = \frac{IC \times CPI_{ORIGINAL} \times T_{CLK}}{IC \times CPI_{NEW\_FP} \times T_{CLK}} = \frac{CPI_{ORIGINAL}}{CPI_{NEW\_FP}} = \frac{2}{1.623} = 1.23$$

Druhá alternativa dává o něco lepší výkonnost.

Poznámka: toto je stejný výsledek jako ve 2. příkladě spočítaném Amdahlovým zákonem.

Přesnější CPU výkonnostní rovnice 6

Souvislost se zadáním 2. příkladu spočítaném Amdahlovým zákonem:

Instrukce	ic_i/IC	cpi_i	$(ic_i/IC) \times cpi_i$	%čas
FP	0.25	4	1	50%
Ostatní	0.75	1.33	1	50%
CPI			2	
FPSQRT	0.02	20	0.4	20% ← protože 0.4 je 20% ze 2

Ve 2. příkladě spočítaném Amdahlovým zákonem je hodnota F_E pro FP instrukce rovna 0.5 (50% doby výpočtu) a pro FPSQRT 0.2 (20% doby výpočtu).

Hodnota S_E pro FP je (původní *cpi* FP instrukcí) / (nové *cpi* FP instrukcí), a to je rovno: $4.0/2.5 = 1.6$, a hodnota S_E pro FPSQRT je potom analogicky rovna: 20/2=10.

Amdahlův zákon vs. CPU výkonnostní rovnice

Výhody použití CPU výkonnostních rovnic pro měření výkonu jsou:

- měřitelný nebo simulací získaný počet instrukcí ic_i
- měřitelné nebo simulací získané cpi_i instrukcí
- oddělený výpočet $ic_i \times cpi_i$ pro zvolenou instrukci, nebo skupinu instrukcí
- výpočet výkonnosti bez znalosti celkové doby provádění vybraných instrukcí

Výhody použití Amdahlova zákona pro měření výkonu jsou:

- jednoduchý výpočet za předpokladu znalosti doby výpočtu vylepšené části a jejího původního podílu na celkovém výpočtu
- výpočet výkonnosti bez znalosti $cpi_i, ic_i, T_{CLK}, CPI$ a IC

Zkušební úlohy: benchmarks 1

Pro měření a vyhodnocení výkonností počítače jsou používané zkušební programy, které se dělí do 5 základních skupin:

1. Reálné aplikace: překladače C, Word, Photoshop atd. Zde existuje problém přenositelnosti, tj. závislosti na OS nebo překladači.
2. Upravené aplikace: reálné aplikace jsou stavebními bloky pro zkušební úlohy. Důvod modifikace je buď zlepšit přenositelnost aplikace, nebo zaměření na určitou část výkonnosti systému.
3. Jádra (kernels): malé klíčové části reálných aplikací (např. Linpack). Nezahrnují vliv paměťového systému (vejdou se do vnitřních skrytých pamětí). Jsou pro uživatele nedostupné.
4. „Toy“ zkušební úlohy: typické 10 – 100 řádkové programy (Eratostenovo síto, Quicksort, atd.). Nezahrnují vliv paměťového systému (vejdou se do vnitřních skrytých pamětí). Uživatel dopředu pozná výstup.
5. Umělé zkušební úlohy: podobné filozofii jader. Zkouší nalézt průměrné hodnoty výskytu operací a operandů v rozsáhlých programech (např. Whetstone a Dhrystone).

Zkušební úlohy: benchmarks 2

Eliminace „slabin“ jedné zkušební úlohy jinou vedla k vytvoření sad zkušebních úloh určených pro různé aplikační oblasti.
Standardní sadou aplikačních zkušebních úloh je sada SPEC (Standard Performance Evaluation Corporation) zkušebních úloh, která se dělí na :

1. Zkušební úlohy pro osobní, stolní a nepřenosné počítače (Desktop benchmarks)
   - CPU (SPEC89, SPEC92, SPEC92, SPEC CPU2000): celočíselné a FP zkušební úlohy, modifikované pro přenositelnost a minimalizování I/O
   - grafické (zahrnují také CPU aktivity): SPECviewperf a SPECapc, podpora OpenGL a CAD/CAM aplikaci
2. Zkušební úlohy pro servery (Server benchmarks)
   - pro měření výkonnosti: CPU propustně orientované zkušební úlohy
   - zkušební úlohy SPEC CPU použité pro víceprocesorový systém

Zkušební úlohy: benchmarks 3

I/O aktivita pro databázové servery a WEB servery

- zkušební úlohy SPECFS (file server)
- zkušební úlohy SPECWeb

Transakční procesy: zkušební úlohy TPC (Transaction Process Council).

3. Zkušební úlohy pro vestavěné počítačové systémy (Embedded benchmarks)
   - nová třída zkušebních úloh, hodnotí se také spotřeba, cena atd.
   - nejlépe standardizovanými zkušebními úlohami se jeví sada EDN Embedded Microprocessor Benchmark Consortium (EEMBC – vysloveno „embassy“), obsahující 5 podskupin:
   1. automobilové/průmyslové
   2. zákaznické
   3. síťové
   4. automatizační pro kancelářskou práci
   5. telekomunikační

Zkušební úlohy: benchmarks 4

Zkušební úlohy pro OS MS Windows

- Business Winstone
  - aplikace sady „office“ (Microsoft, Corel, WordPerfect)
  - skripty simulující uživatele přepínajícího a spouštějícího množství aplikací
- CC Winstone
  - práce s větším objemem dat (Photoshop, Premiere a různé audio editovací programy)
  - skripty simulující uživatele spouštějícího skupinu smíšených aplikací
- Winbench
  - skripty spouštějící testy výkonnosti CPU video systémů, disků atd.
  - používají se jádra se zaměřením na každý podsystém

Zkušební úlohy: benchmarks 5

LINPACK – [MFLOPS] FP zkušební úloha

- řešení SLR ($\mathbf{Ax} = \mathbf{b},\ hod(\mathbf{A}) = 100$)
- mnoho FP operací, ale použito jen několik typů
- vzhledem k tomu, že operace $y_i = y_i + a \cdot x_i$ jsou prováděny nejdelší dobu, může se projevit silný vliv i malé, instrukční vnitřní skryté paměti
- data jsou rozmístěna ve velkém prostoru
- vhodná jako zkušební úloha pro vektorové a paralelní superpočítače
- verze LINPACK:
  - single/double
  - rolled/unrolled

Zkušební úlohy: benchmarks 6

Starší známé zkušební úlohy (jejich použití se dnes nedoporučuje)

- Whetstone – [Whips]
  - FP výpočty nad velkým množstvím dat, také celočíselné výpočty, lokálních proměnných málo, většina globálních proměnných (skalární a jednorozměrná pole)
  - výsledný kód malý a může se vejít do vnitřní skryté paměti
  - v úloze využity transcendentní funkce, podmíněné skoky, volání procedur a indexování polí
  - určená k měření propustnosti vědeckotechnických výpočtů, 963 tisíc instrukcí
- Dhrystone – [Dhrystone]
  - typická sada celočíselných výpočtů, neobsahují FP operace
  - většina proměnných je lokální
  - je určena pro systémové programování, není vhodná z hlediska numerických výpočtů

Měření výkonností - MIPS, MFLOPS

Definice měření výkonnosti jednotkami MIPS

$$MIPS = \frac{IC}{T_{CPU} \times 10^6} = \frac{IC}{IC \times CPI \times T_{CLK} \times 10^6} = \frac{f_{CLK}}{CPI \times 10^6} = [10^6 \text{ instr.}/s]$$

MIPS závisí na:

- ISA a programu, neboť CPI závisí na programu, mixu instrukcí

Důsledek:

- výkonnost vyjádřena v jednotkách MIPS je závislá na programu, i když se tato závislost často neuvádí

Měření výkonnosti MFLOPS – totéž pro FP instrukce

- k vyjádření špičkové výkonnosti superpočítačů, reálná výkonnost může být značně nižší