Architektura počítačů
Výkonnost počítačů

http://d3s.mff.cuni.cz/teaching/computer_architecture/

Lubomír Bulej

bulej@d3s.mff.cuni.cz

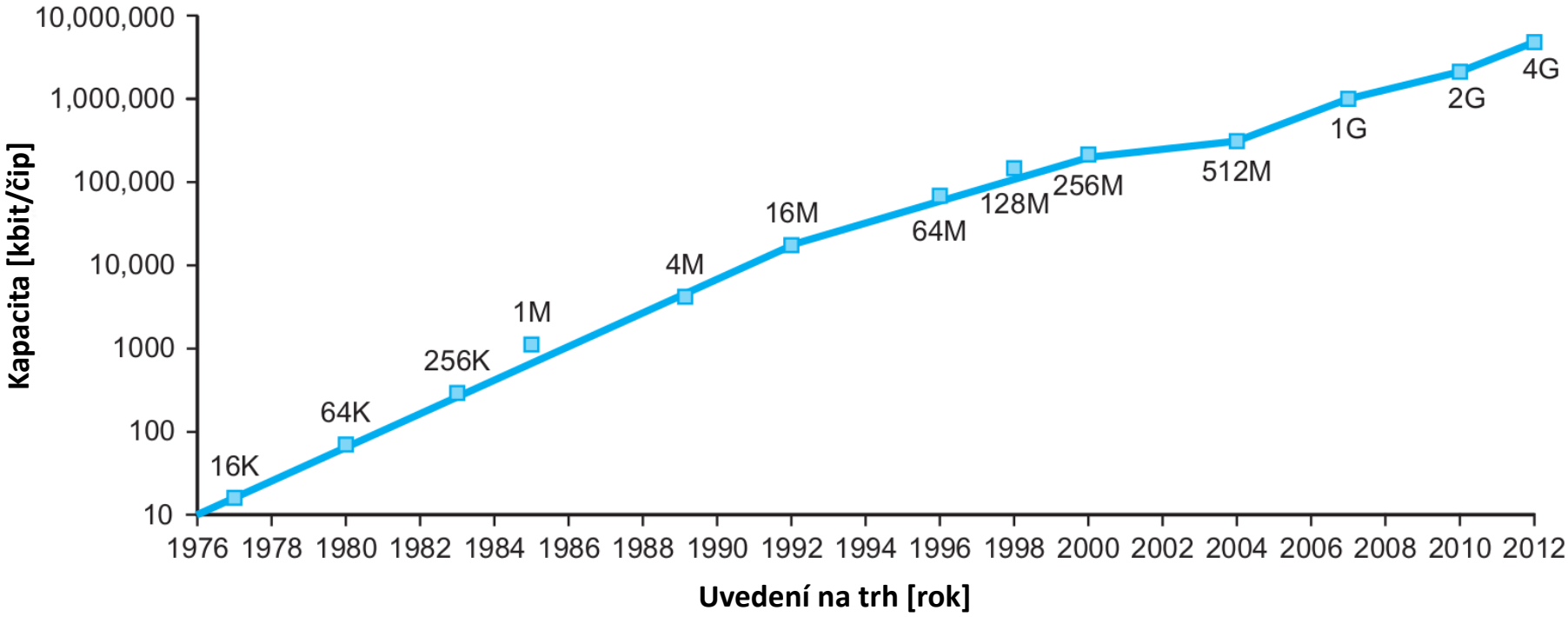
CHARLES UNIVERSITY IN PRAGUE

faculty of mathematics and physics

Růst výkonnosti stavebních prvků

Rok	Technologie	Relativní výkon / jednotková cena
1951	Elektronka	1
1965	Tranzistor	35
1975	Integrovaný obvod (nízká integrace)	900
1995	Integrovaný obvod (velmi vysoká integrace)	2 400 000
2005	Integrovaný obvod (ultra vysoká integrace)	6 200 000 000

Růst kapacity DRAM čipů



Source: P&H

„Big ideas“ v architektuře

- Design pro Moorův zákon
- Abstrakce (pro zjednodušení návrhu)
- Optimalizace pro běžný případ
- Paralelismus
- Pipelining
- Predikce a spekulace
- Hierarchie pamětí
- Redundance (spolehlivost)

Moorův „zákon“

- **Gordon Moore (*1929)**
  - Zakladatel společnosti Intel
  - **Předpověď:** Množství tranzistorů integrovaných na jednom čipu se zdvojnásobí každých 18 – 24 měsíců
    - 60. léta 20. století
    - Zmenšování velikosti vedoucí k růstu teoretické rychlosti a kapacity
    - Často se vztahuje i na jiné oblasti
      - Disková kapacita, přenosové pásmo

Moorův „zákon“ (2)

- **Exponenciální růst posledních 40 let!**
  - Udržování Moorova „zákona“ v platnosti vyžaduje neustálý **velmi netriviální** technologický pokrok
    - Zatím víceméně v rámci jedné technologické domény (polovodičové tranzistory)
    - Nepřekonatelné fyzikální limity (kvantový tunelový efekt, odvod odpadního tepla, kvantový šum)
  - Stále častěji nutné kompromisy
    - Počet tranzistorů není v přímé úměře k hrubému výpočetnímu výkonu pro sekvenční algoritmy

Výkonnost programu (1)

Faktor	Způsob ovlivnění
Algoritmus	Počet základních operací a I/O operací jako funkce velikosti vstupu
Programovací jazyk, překladač, architektura počítače	Počet instrukcí na každou základní operaci
Procesor, paměť	Rychlost provádění instrukcí
I/O subsystém (hardware, operační systém)	Rychlost provádění I/O operací

Proč se zabývat výkonností?

- **Porovnávání počítačů**
  - Vyhrává levnější a/nebo lepší produkt
    - Oblast osobních počítačů: tvrdá konkurence
    - Oblast vestavěných počítačů: optimalizace ceny výsledného produktu
  - Důležité pro kupující → důležité pro návrháře a výrobce
- **Ověření přínosu změn v architektuře**
  - Systematické zkoumání vlivu architektury na výkon
    - Jediná indikace, zda je pokrok skutečně pokrokem

Co je vlastně výkonnost?

- **Počítač A je „lepší“ než počítač B**
  - Co to vlastně znamená? Lepší v čem?
  - Je nákladní vůz „lepší“ než sportovní vůz?
  - Je Concorde „lepší“ letadlo než Boeing 747?

Letadlo	Kapacita [osoby]	Dolet [km]	Cestovní rychlost [km/h]	Propustnost [osob·km/h]
Boeing 777	375	9000	905	339375
Boeing 747	470	7700	905	425350
Concorde	132	7400	2158	284856
Douglas DC-8-50	146	16000	810	118260

Co je vlastně výkonnost? (2)

- **Základní kritéria výkonnosti**
  - Kvalifikace výkonnosti
  - Co porovnáváme
  - Co potřebujeme
- **Základní měřítka výkonnosti**
  - Též „metriky výkonnosti“
  - Kvantifikace výkonnosti
  - Vzhledem k pevně daným kritériím výkonnosti udávají způsob, jak výkonnost měřit a porovnávat

Základní metriky výkonnosti

- **Doba běhu (*execution time*)**
  - Čas vykonávání konkrétní úlohy
  - Zajímavé pro konkrétního uživatele
- **Doba odezvy (*response time*)**
  - Čas potřebný k získání výsledku dané úlohy
  - Důležité především pro interaktivní úlohy
- **Propustnost (*thoughput*)**
  - Množství práce vykonané za jednotku času
  - Zajímavé pro správce zdroje (serveru, datového centra)

Výkonnost z pohledu uživatele

- **Celkový čas**
  - *Wall-clock time, reponse time, elapsed time*
  - Zahrnuje čekání na I/O operace, režii OS apod.
    - Včetně sdílení zdrojů s jinými uživateli
  - Výkonnost systému jako celku
- **Procesorový čas**
  - *CPU execution time, CPU time*
  - Čas, po který procesor skutečně vykonával program
    - Nezahrnuje čekání na I/O operace a čas, kdy program fakticky neběžel
    - Zahrnuje režii OS
      - **Uživatelský** a **systémový** čas procesoru
  - Výkonnost procesoru

Výkonnost z pohledu návrháře

- **Rychlost provádění instrukcí**
  - Frekvence hodin (*clock rate*)
    - Taktovací frekvence, frekvence hodinového signálu apod.
  - Délka hodinového cyklu (*clock cycle*)
    - Perioda hodinového signálu apod.

$$CPU\ execution\ time = \frac{CPU\ clock\ cycles}{CPU\ clock\ rate}$$

$$CPU\ execution\ time = CPU\ clock\ cycles \times CPU\ clock\ cycle\ time$$

Výkonnost z pohledu překladače

- **Průměrný počet taktů na vykonání instrukce**
  - *Clock cycles per instruction* (CPI)
  - Vztaženo ke konkrétnímu programu nebo jeho části
  - Umožňuje srovnávat různé implementace stejné architektury
    - Při zachování počtu instrukcí na program

$$CPU\ clock\ cycles = CPI \times Number\ of\ instructions$$

Základní vztah pro výkonnost procesoru

- **Vztah počtu instrukcí, CPI a délky cyklu**
  - 3 různé faktory ovlivňující výkonnost
    - Srovnání různých implementací stejné architektury
    - Zhodnocení alternativní architektury

$$CPU\ execution\ time = CPI \times Number\ of\ instructions \times CPU\ clock\ cycle\ time$$

$$CPU\ execution\ time = \frac{CPI \times Number\ of\ instructions}{CPU\ clock\ rate}$$

Výkonnost procesoru (1)

- **Výkonnost procesoru při vykonávání daného programu**
  - Závisí na počtu instrukcí, průměrném počtu taktů na instrukci (CPI), délce hodinového cyklu (taktovací frekvenci)
  - **Žádný z faktorů sám o sobě není vypovídající** ☠
    - Snížení počtu instrukcí → architektura s nižší taktovací frekvencí nebo vyšším CPI
    - CPI závisí na **instrukčním mixu** (četnost a typ prováděných instrukcí) daného programu
      - Kód s nejmenším počtem instrukcí nemusí být nejrychlejší

Výkonnost procesoru (2)

- **Výkonnost procesoru při vykonávání daného programu**
  - Jediným úplným a spolehlivým měřítkem je empiricky změřený reálný procesorový čas
    - Není však vypovídající o procesorových časech jiných programů

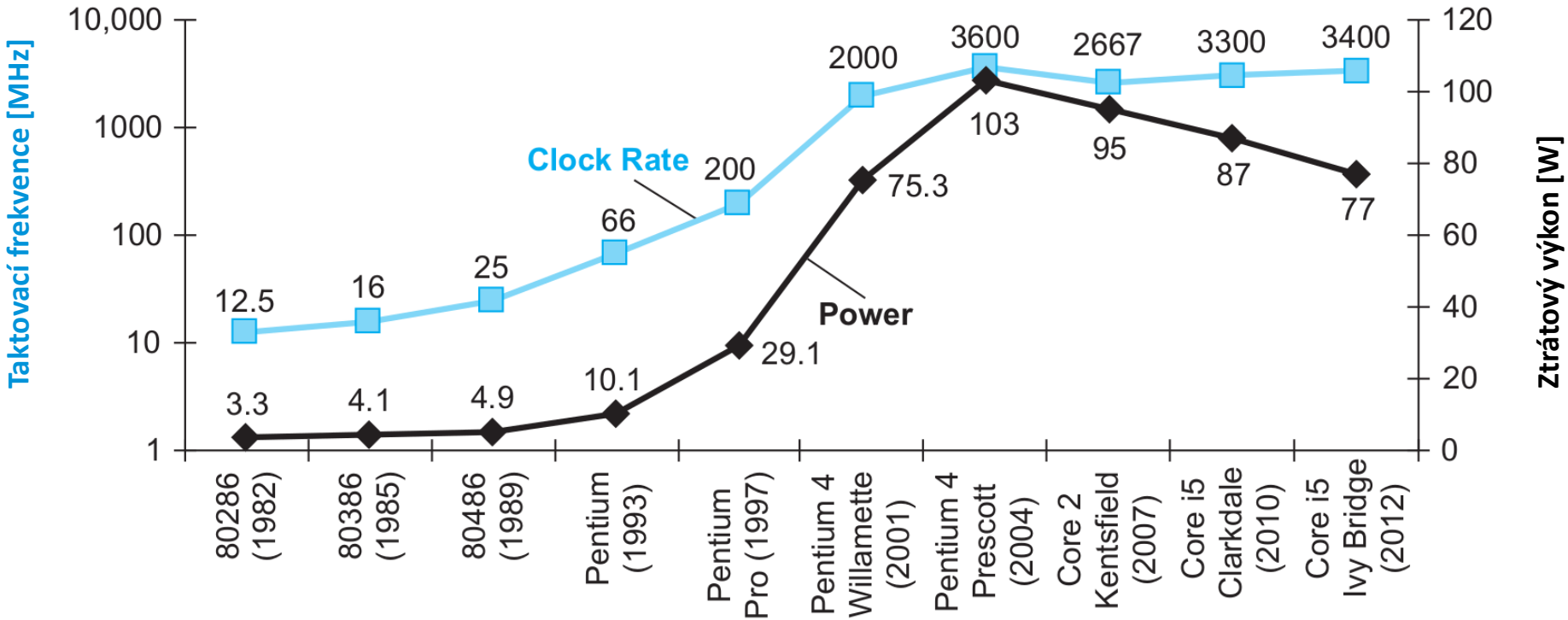
Výkonnost programu (2)

Faktor	Ovlivnění	Způsob ovlivnění
Algoritmus	Počet instrukcí CPI	Počet základních operací Datové typy (celočíselné vs. neceločíselné)
Programovací jazyk	Počet instrukcí CPI	Typ základních operací Abstraktní datové typy
Překladač	Počet instrukcí CPI	Způsob překladu základních operací na instrukce
Architektura	Počet instrukcí CPI Taktovací frekvence	Instrukce potřebné k vyjádření algoritmu Technologie (maximální taktovací frekvence) Doba provádění instrukcí v taktech

Výkonnost programu (3)

Konec zlatých časů

Taktovací frekvence a ztrátový výkon



Source: P&H

Ztrátový výkon

- ***Complementary Metal Oxide Semiconductor (CMOS)***
  - Převládající technologie výroby integrovaných obvodů
  - Minimální spotřeba v klidovém stavu
  - Dynamický ztrátový výkon
    - Kapacitní zátěž (vodiče, tranzistory, zátěž na výstupu)
    - Provozní napětí (ovlivňuje rychlost přepínání)
    - Frekvence přepínání (odvozená od taktovací frekvence)

$$P \approx \frac{1}{2} \times C \times U^2 \times f$$

Ztrátový výkon (2)

- **Reálné důsledky**
  - Za posledních 20 let
    - Přibližně 1000násobný nárůst frekvence přepínání
    - Přibližně 30násobný nárůst ztrátového výkonu
    - Nutný pokles provozního napětí z 5 V na 1 V
      - 15 % pokles s každou generací
      - Fyzikální limity (energetická zeď)
- **Příklad**
  - Snížením kapacitní zátěže, provozního napětí a pracovní frekvence o 15% dostaneme procesor s cca 52% spotřebou oproti výchozímu stavu.

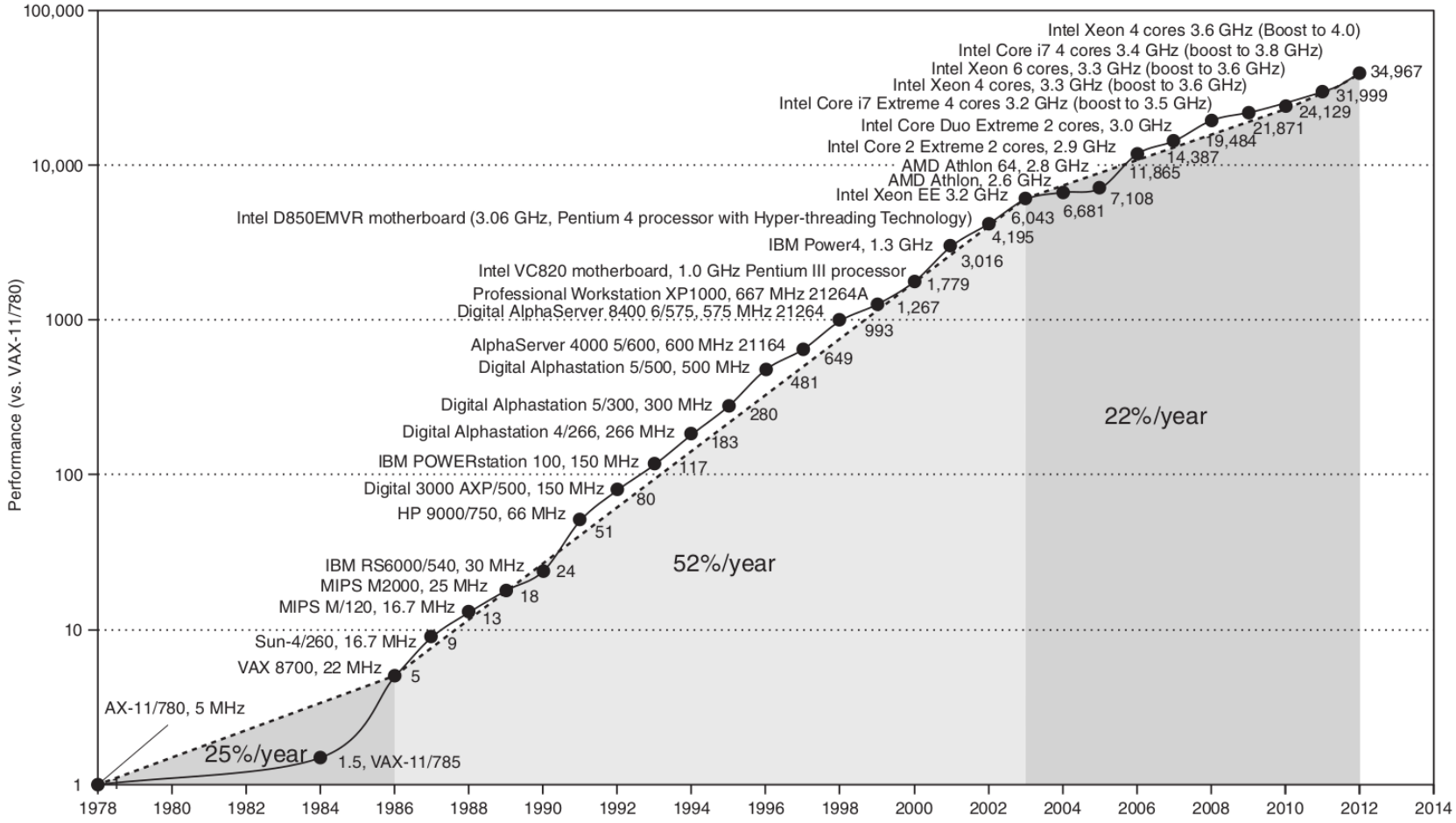
Energetická zeď

- **Napětí není možné neomezeně snižovat**
  - Problém spolehlivosti (odstup signál/šum)
  - Projev statických ztrát (tunelování elektronů apod.)
    - Asi 40 % spotřeby dnešních procesorů
- **Chlazení není snadné jednoduše zlepšovat**
  - Malá plocha procesorového čipu
  - Vypínání nepoužívaných částí čipu v daném taktu
  - Vodní a jiné chlazení není příliš praktické

Energetická zeď (2)

- **Nové cesty ke zvyšování výkonu**
  - Odlišné od těch, které se používají posledních 30 let
  - **Přechod od jednoprocesorových k víceprocesorovým architekturám**

Nárůst výkonu osobních počítačů



Source: P&H

Víceprocesorové systémy

- **Dříve**
  - Více samostatných procesorů (*multiprocessor*)
  - Superpočítače, high-end servery
  - U osobních a vestavěných počítačů raritou
- **Nyní**
  - Moorův „zákon“ stále platí (stále se daří zlepšovat technologii pro zdvojnásobování počtu tranzistorů)
  - Výroba procesorů obsahující více samostatných procesorových jader (*multicore*)

Vícejádrové systémy

- **Vliv na výkon systému**
  - Zvyšování propustnosti (*throughput*)
    - Zpracování více paralelních požadavků současně
  - Taktovací frekvence a CPI zůstávají stejné
    - Rychlost sekvenčních algoritmů se nemění
- **Důsledky pro programátory**
  - Programy se (samy) nezrychlí díky novým technologiím
    - Místy pomohou lepší API
  - Je nutné začít využívat více procesorů/jader explicitně
    - Každých 18 – 24 měsíců se jejich počet zdvojnásobí

Proč tolik povyku?

- **Fundamentální změna rozhraní HW/SW**
  - Paralelismus byl vždy důležitý, ale před softwarem byl skrytý
    - Paralelismus na úrovni instrukcí (*instruction-level parallelism*), zřetězené zpracování instrukcí a další techniky
    - Programátor i překladač vytvářeli sekvenční kód
  - Nyní je potřeba psát programy explicitně paralelní
- **Experimenty s paralelními architekturami probíhají již nejméně 40 let**
  - Vždy však šlo o slepé větve vývoje (nedošlo k masovému rozšíření, protože programátoři nové paradigma nepřijali)
  - Nyní celý IT průmysl vsadil na tuto kartu

Proč je paralelní programování těžké?

- **Programování zaměřené na výkon**
  - Vedle správné funkce je důležitá také rychlost
    - *„Pokud nám nejde o rychlost, můžeme dál programovat sekvenčně.“* (P&H)
  - Lidé uvažují „sekvenčně“ a „jednovláknově“
  - Plánování a koordinace práce více procesorů nesmí mít vysokou režii

Proč je paralelní programování těžké? (2)

- **Analogie z reálného života**
  - 1 redaktor napíše 1 článek za 2 hodiny
  - 8 redaktorů napíše 1 článek za *X* hodin
    - Ideální paralelní zpracování: *X* = 15 minut
    - Reálné problémy
      - Plánování *(scheduling)*
      - Vyvažování zátěže *(load balancing)*
      - Režie na komunikaci a synchronizaci *(communication and synchronization overhead)*

Měření výkonnosti

- **Porovnávání výkonnosti různých počítačů**
  - Pro konkrétní program už umíme (reálný procesorový čas)
  - Víme, že porovnávání jednotlivých faktorů (taktovací frekvence, CPI, počet instrukcí) není vypovídající pro jiné programy
  - Jak definovat obecnější metriku, která by dovedla aproximovat výkonnost zpracování nějaké větší množiny programů?

Měření výkonnosti (2)

- **Zátěž (*workload*)**
  - Typická pracovní zátěž počítače (sada typicky spouštěných programů)
  - Složité na definování, složité na automatizaci (opakované a opakovatelné spouštění)
- **Výkonnostní test (*benchmark*)**
  - Program specificky určený k měření výkonnosti
  - Sada benchmarků
    - Statisticky relevantní reprezentace typické pracovní zátěže
    - Doufáme, že měření benchmarků předpoví chování počítače při skutečné zátěži stejného nebo podobného charakteru

Měření výkonnosti (3)

- **SPEC (Standard Performance Evaluation Corporation)**
  - Sdružení komerčních a nekomerčních subjektů
    - Výrobci procesorů, počítačů, překladačů, operačních systémů apod.
    - Vědecké instituce
  - **Cíl:** Definice vhodných sad benchmarků pro jednoduché porovnávání výkonnosti
    - Různé sady benchmarků pro různé workloady
    - Původně hlavně výkonnost CPU, dnes i GPU, překladačů, databází, emailových systémů atd.

SPEC CPU 2006

- **Měření výkonnosti procesoru**
  - CINT2006 (celočíselné výpočty)
    - 12 benchmarků (překladač C, šachový algoritmus, simulace kvantového počítače atd.)
  - CFP2006 (výpočty v plovoucí řádové čárce)
    - 17 benchmarků (metoda konečných prvků, molekulární dynamika atd.)
  - SPECratio
    - Podíl referenční a změřené doby běhu benchmarku
    - Souhrnné hodnocení: Geometrický průměr v dané sadě

$$\sqrt[n]{\prod_{i=1}^{n} SPECratio_i}$$

SPEC CINT2006 pro AMD Opteron X4

Description	Name	Instruction Count × 10^9	CPI	Clock cycle time (seconds × 10^9)	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2,118	0.75	0.4	637	9,770	15.3
Block-sorting compression	bzip2	2,389	0.85	0.4	817	9,650	11.8
GNU C compiler	gcc	1,050	1.72	0.4	724	8,050	11.1
Combinatorial optimization	mcf	336	10.00	0.4	1,345	9,120	6.8
Go game (AI)	go	1,658	1.09	0.4	721	10,490	14.6
Search gene sequence	hmmer	2,783	0.80	0.4	890	9,330	10.5
Chess game (AI)	sjeng	2,176	0.96	0.4	837	12,100	14.5
Quantum computer simulation	libquantum	1,623	1.61	0.4	1,047	20,720	19.8
Video compression	h264avc	3,102	0.80	0.4	993	22,130	22.3
Discrete event simulation library	omnetpp	587	2.94	0.4	690	6,250	9.1
Games/path finding	astar	1,082	1.79	0.4	773	7,020	9.1
XML parsing	xalancbmk	1,058	2.70	0.4	1,143	6,900	6.0
Geometric Mean							11.7

Source: P&H

SPEC CINT2006 pro Intel Core i7 920

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

Source: P&H

Bludy a pasti

Marná očekávání

- **Při zlepšení výkonnosti části systému dojde k úměrnému zlepšení výkonnosti celého systému**
  - **Celková délka běhu:** 100 s
    - **Z toho instrukce pro násobení:** 80 s
  - Kolikrát je potřeba zrychlit násobení, aby se celkově program zrychlil 5×?

Marná očekávání (2)

$$Execution_{fast} = \frac{Execution_{slow}}{5}$$

$$Execution_{slow} = 80 + 20$$

$$Execution_{fast} = \frac{80}{n} + 20$$

$$\frac{80}{n} + 20 = \frac{80 + 20}{5}$$

$$\frac{80}{n} + 20 = 20$$

$$\frac{80}{n} = 0$$

$$80 \neq 0$$

Amdahlův zákon

- **Gene Amdahl (* 1922)**
  - Různá znění Amdahlova zákona
    - Nejobecnější pro teoretické zrychlení sekvenčního algoritmu s využitím více vláken (formulováno v roce 1967)
    - Častá varianta jako reformulace zákona klesajících výnosů
      - ***Zvýšení výkonnosti dosažitelné nějakým zlepšením je omezené mírou používání tohoto zlepšení***

[1]

$$Speedup(n) = \frac{1}{B + \frac{1}{n}(1 - B)} \qquad n \in \mathbb{N} \qquad B \in \langle 0, 1 \rangle$$

Amdahlův zákon (2)

- **Praktické důsledky**
  - *Make the common case fast*
    Optimize for the common case
  - Optimalizace má největší užitek pro nejčastější případy
    - Nejčastější případy jsou často mnohem jednodušší než speciální případy, takže se lépe optimalizují
  - I výrazné optimalizace speciálních případů mohou přinést menší užitek než drobné optimalizace nejčastějších případů

Špatná měřítka výkonnosti

- **Použití podmnožiny faktorů (taktovací frekvence, CPI, počet instrukcí) jako samostatné metriky**
  - Použití jen jednoho faktoru je téměř vždy špatně
  - Použití dvou faktorů může být ve specifických případech v pořádku, ale často tomu tak není
    - Jednotlivé faktory existuje závislost
  - Jiné metriky, které jsou jen převlečené známé faktory

Špatná měřítka výkonnosti (2)

- **MIPS (Million Instructions Per Second)**
  - Rychlost vykonávání instrukcí
  - Intuitivní (čím vyšší, tím rychlejší)
  - Problémy
    - Nebere v úvahu možnosti instrukcí, dobu vykonávání jednotlivých instrukcí atd.
      - Nelze porovnávat počítače s různou instrukční sadou
    - Liší se podle konkrétního instrukčního mixu daného programu (jedna hodnota nereprezentuje výkon počítače)

$$MIPS = \frac{Instruction\,count}{10^6 \times Execution\,time}$$

Reference

[1] http://upload.wikimedia.org/wikipedia/commons/7/79/Amdahl_march_13_2008.jpg