INTRODUCTION / PERFORMANCE

Computer architecture: @ the boundary of hardware & software

- → parallelism
- → measuring & evaluation
- → operating systems
- → instruction set architecture
- → history
- → compiler design
- → programming languages

DEFINITION: COMPUTER ARCHITECTURE

→ the science and art of selecting and interconnecting hardware components to create computers that meet functional, performance, and cost goals

---

Performance

→ Latency - time to finish a fixed task (execution time)

→ Throughput - # tasks per unit time (bandwidth)

↳ exploit parallelism for throughput ← ~~not latency~~

Speedup

Speedup of A over B

→ X = latency(B) / latency(A) ← faster in denominator)

→ X = throughput(A) / throughput(B) ← slower in denom.

→ example: ↳ car capacity: 5 ppl speed: 60 mi/hr

↳ bus capacity: 60 ppl speed: 20 mi/hr

↳ latency: car is 3x (200%) faster than bus

↳ throughput: bus is 4x (300%) faster than car

Comparing performance

→ Given: program A ← 200 cycles
program B ← 350 cycles

→ speedup of A over B?

↳ speedup = 350/200 = 1.75 (= 75%)

→ If program C is 50% faster than A, how many cycles does C run for? $200/X = 1.5 \Rightarrow X = \frac{200}{1.5} = 133$ cycles

---

Speedup of A over B

What if $X < 1$?
→ A is slower than B

$X = \text{latency}(B)/\text{latency}(A)$

$X = \text{throughput}(A)/\text{throughput}(B)$

Percent increase and decrease are **not** the same.

→ % increase of cycles $= \left(\frac{350-200}{200}\right) \cdot 100 = 75\%$

→ % decrease of cycles $= \frac{350-200}{350} \cdot 100 = 42.3\%$

AVERAGING PERFORMANCE

Arithmetic: $(1/N) \cdot \sum_{P=1..N} \text{latency}(P)$ ← for units proportional to time (e.g. latency)

Harmonic: $\frac{N}{\sum_{P=1..N} 1/\text{throughput}(P)}$ ← for units inversely proportional to time (e.g. throughput)

Geometric: $\sqrt[N]{\prod_{P=1..N}}$ ← for unitless quantities (e.g. speedup ratios)

You can add latencies, not throughputs

→ latency (p1+p2, A) = latency (p1, A) + latency (p2, A)

→ throughput (p1+p2, A) != throughput (p1, A) + throughput (p2, A)

Example: you drive 1 mi @ 30 mph + 1 mi @ 90 mph average speed?

unit - inversely proportional to time ⇒ harmonic mean

$$\text{mean} = \frac{N}{\Sigma\ 1/30 + 1/90} = \frac{2}{4/90} = \frac{2 \cdot 90}{4} = \boxed{45 \text{ mph}} \neq \frac{30+90}{2}$$

CPU Performance Equation

→ CPI impacted by program, compiler, ISA, micro arch

$$\text{latency} = \frac{\text{seconds}}{\text{program}} = \frac{\text{instr}}{\text{program}} \cdot \frac{\text{cycles}}{\text{instr}} \cdot \frac{\text{seconds}}{\text{cycle}}$$

→ instruction count

→ impacted by program, compiler, ISA

↳ clock period (Hz)

→ impacted by micro-arch, technology

→ minimize all 3 for low latency (better performance)

---

cycles per instruction = CPI

→ $\text{CPI} = \frac{\text{cycles}}{\text{instr}}$

↳ IPC = 1/CPI

↳ different instr. have diff. cycle costs

→ example

a program executes equal # of 3 diff instr.

inst	cycles/instr
int	1
mem	2
FP	3

FIND CPI

use arithmetic mean

$\frac{1+2+3}{3} = 2 \text{ cycles/instr}$

other performance metrics

→ MIPS ← $\frac{\text{millions instr.}}{\text{sec}}$

→ FLOPS ← # floating pt instr/sec

CPI example assume a processor w/ instruction frequencies & cost

instr	freq	cost (cycles)
ALU	50%	1
load	20%	5
store	10%	1
branch	20%	2

Which change would improve performance more?

(A) branch prediction which reduce branch cost to 1 cycle

(B) faster data memory to reduce load cost to 3 cycles

base: $1 \cdot 0.5 + 5 \cdot 0.2 + 1 \cdot 0.1 + 2 \cdot 0.2 = 2$ CPI

B: $3 \cdot 0.2$

A: $1 \cdot 0.2 = 1.8$ CPI

$= 1.6$ CPI ← B is fastest

Example average MIPS for M1 & M2? clock: 1GHz

instr	freq	CPI_M1	CPI_M2
ALU	50%	1	2
branch	15%	2	1
load	20%	2	1
store	15%	1	1

① find IPC $= \frac{\text{instr}}{\text{cycle}}$

② MIPS $= \frac{\text{instr}}{\text{cycle}} \cdot \frac{\text{cycle}}{\text{sec}} \cdot 1E\text{-}6$

$0.5 + 0.15(2) + 0.2(2) + 0.15(1) = 1.35$ M1 $\frac{\text{cycles}}{\text{instr}}$

$0.5(2) + 0.15 + 0.2 + 0.15 = 1.5$ M2 $\frac{\text{cycles}}{\text{instr}}$

MIPS M1 $= \frac{1}{1.35} \frac{\text{instr}}{\text{cycle}} \cdot \frac{\text{cycle}}{1E\text{-}9 \text{ sec}} =$ 741 MIPS

MIPS M2 $= \frac{1}{1.5} \frac{\text{instr}}{\text{cycle}} \cdot \frac{\text{cycle}}{1E\text{-}9 \text{ sec}} =$ 667 MIPS

Peak MIPS, CPI = 1 = IPC

$$\frac{1}{1E\text{-}9} \cdot \frac{1}{E\,6} = \frac{1}{1E\text{-}3} = 1E3 = \boxed{1000 \text{ MIPS}}$$

Amdahl's Law

$$\frac{1}{(1-P) + \frac{P}{S}}$$

P = proportion of running time affected by optimization

S = speedup

example: speedup 25% of a program's execution by 2x?

P = 0.25, S = 2

$$\frac{1}{(0.75) + \frac{0.25}{2}} = \boxed{1.14x}$$

example: P = 25% S = ∞

$$\frac{1}{0.75 + \frac{0.25}{\infty}} = \boxed{1.33x}$$

Amdahl's Law for Parallelization

$$\frac{1}{(1-P) + \frac{P}{N}}$$

How much will parallelization improve performance?

P = portion of parallel code

N = threads

example: What's the max speedup for a program thats 10% serial? (N = ∞) (=> P = 0.9)

$$\frac{1}{(1-0.9) + \frac{0.9}{\infty}} = \frac{1}{0.1} = \boxed{10x} \text{ speedup}$$

example: max speedup, 1% serial?

$$\frac{1}{1-0.99} = \frac{1}{0.01} = \boxed{100x} \text{ speedup}$$

If you could pick one type of instr to make twice as fast, which instr type would you pick?

another example

program has:

instr	latency	freq
mem	4	40%
add	2	50%
mult	16	10%

mem : $4 \cdot 0.4 = 1.6$ ←
add : $2 \cdot 0.5 = 1$
mult : $16 \cdot 0.1 = 1.6$ ←
choose mem or mult

Performance must be associated w/a workload ← set of tasks

benchmarks: standard workloads, representative of real programs

micro-benchmarks: non-standard workloads, tiny programs used to evaluate specific aspects of performance

(e.g. binary tree search, towers of hanoi, 8-queens, etc)
