Lecture 3: Performance

Sunday, January 14, 2018 7:21 AM

Outline

- Latency vs. bandwidth
- Benchmarks
- Iron law
- Speedup
- Amdahl's law

Latency vs bandwidth: Moving data around

[Grace Hopper - Nanoseconds](https://youtu.be/JEpsKnWZrJ8)

SeHouMusic

https://youtu.be/JEpsKnWZrJ8

Nanosecond → latency →

Goodness of a system?

Transfer rate → bandwidth

Inst. per second (IPS) → throughput →

Accuracy Security Usability

Storage space ☆ Price Stability

Energy Size

3yr

2/wk

~ 300 bottles

Little's Law

Little's Law

Resources = bandwidth * latency

2/wk ↑ ↔
100/yr ↓

$3 \times 100 = 300$ bottles

Measure a system

Benchmarking
- → tests → fast
- → standard
- → "score"
- → realistic

Geekbench
SPEC →
Kraken

↳ predict real-world performance/other metric

Problems

Rarely representative of real world
Not full applications

☆ take results with a grain of salt

Iron Law of performance

Simple model

Application: 1 million inst
System: 500 MHz
5 cycles per instruction (CPI)

$$\text{Time} = \text{Inst} \cdot \frac{\text{cycles}}{\text{inst.}} \cdot \frac{s}{\text{cycle}}$$

Inst · CPI · Cycle time ↳ $\frac{1}{\text{freq.}}$

$1 \cdot 10^6 \cdot 5 \cdot \frac{1}{500\,\text{MHz}}$

Hz = 1/s

$1\,10^6 \cdot 5 \cdot 2 \cdot 10^{-9}\,s = 10 \cdot 10^{-3} = 10\,ms$

Separate out different design considerations

Time = Inst * CPI * Cycle time → reduce cycle time ⇒ technology, incr. freq.

↳ Change your alg.
↳ change ISA (x86, ARM, MIPS) → Reduced inst set (RISC)
↳ Complex Inst set (CISC)

ld \$r1
add \$r1, \$r2

add (\$rax), \$rbx → Micro-ops

↳ change compiler

to reduce cycles per inst ⇒ improve microarchitecture
pipelining or diff. CPU design

Today's systems CPI ⇒ 1/6 6 inst/cycle
→ 1/2

Latency ⇒ how to compare system A (AMD) system B (Intel)

Latency ⇒ how to compare system A (AMD) system B (Intel)

	A (AMD)	B (Intel)
gcc	2s	1s

Intel is twice as fast for gcc compiling program X

takes 1 less second

half as long

AMD is 50% slower

Speedup → it's the ratio that gives a big number

→ $\frac{\text{time on old system}}{\text{time on new system}}$ $\frac{2s}{1s}$ = Intel has a 2x speedup

Program	A	B	Baseline
30% loads	1x	1x	1x
20% stores	1x	1x	1x
→ 10% divides	5x	1x	1x
40% ALU	1x	1.5x	1x

Majority is ALU

→ **Make the common case fast**

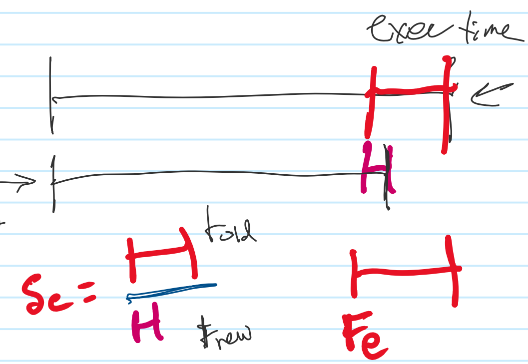
System A speedup = $\frac{T_{old}}{T_{new}} = \frac{1}{.3+.2+.1\cdot\frac{1}{5}+.4} = \frac{1}{.92} = 1.09x$

$\boxed{B} = \frac{1}{.3+.2+.1+\frac{.4}{1.5}} = \frac{1}{.6+\frac{.8}{3}} = 1.15x$

$T_{new} = T_{old} * \left[(1-F_e) + \frac{F_e}{S_e}\right]$

$(1-F_e)$ ↓ fraction enhanced

S_e ↳ speedup of enhancement



Amdahl's Law

Speedup of app. on new system

$\text{Speedup} = \frac{T_{old}}{T_{new}} = \frac{1}{(1-F_e)+\frac{F_e}{S_E}}$

$F_e = 0.5$

Max speedup?

$\frac{1}{1-.5+\frac{.5}{\infty}} = \frac{1}{.5} = 2$