U. Wisconsin CS/ECE 552
Introduction to Computer Architecture

Prof. Karu Sankaralingam

Performance (Chapter 4)

www.cs.wisc.edu/~karu/courses/cs552/

Slides combined and enhanced by Karu Sankaralingam from work by Falsafi, Hill, Marculescu, Nagle, Patterson, Roth, Rutenbar,Schmidt, Shen, Sohi, Sorin, Thottethodi, Vijaykumar, & Wood

Performance of Computers

- Which computer is fastest?
- Not so simple
  - Scientific simulation - FP performance
  - Authoring programs - Integer performance
  - Commercial work - I/O & vast memory

Performance of Computers

- Want
  - Highest Performance (modeling oil fields)
  - Lowest Cost (doorknob)
  - Lowest Cost/Performance (most common)
- Performance will depend on workload
- Computers not completely interchangable
  - PC cannot (currently) have 128 GB memory

Outline

- Time and performance
- Iron law
- Metrics: MIPS and MFLOPS
- Which programs and how to average
- Amdahl's law

Defining Performance

- What is important to who?

1. Computer system user
   - minimize elapsed time for program = time_end - time_start
   - called response time
2. Computer center manager
   - maximize completion rate = #jobs/second
   - called throughput

Response Time vs. Throughput

- Is throughput = 1/av. response time?
  - only if NO overlap
  - with overlap, throughput > 1/av.response time
- e.g., a lunch buffet - assume 5 entrees
  - each person takes 2 minutes at every entree
    - throughput is 1 person every 2 minutes (1/2)
    - BUT time to fill up tray is 10 minutes
  - otherwise, why and what would the throughput be?
    - because there are 5 people (each at 1 entree) simultaneously;
    - if there is no such overlap throughput = 1/10

What is Performance for us?

- For computer architects
  - CPU execution time = time spent running a program
- Because people like faster to be bigger to match intuition
  - performance = 1/X time
  - where X = response, CPU execution, etc.
- Elapsed time = CPU execution time + I/O wait
- We will concentrate mostly on CPU execution time

Improve Performance

- Improve (a) response time or (b) throughput?
  - faster CPU
    - both (a) and (b)
  - Add more CPUs
    - (b) but (a) may be improved due to less queueing

Performance Comparison

- Machine A is n times faster than machine B iff
  - perf(A)/perf(B) = time(B)/time(A) = n
- Machine A is x% faster than machine B iff
  - perf(A)/perf(B) = time(B)/time(A) = 1 + x/100
- E.g., A 10s, B 15s
  - 15/10 = 1.5 => A is 1.5 times faster than B
  - 15/10 = 1 + 50/100 => A is 50% faster than B

Breaking Down Performance

- A program is broken into instructions
  - H/W is aware of instructions, not programs
- At lower level, H/W breaks instructions into cycles
  - lower level state machines change state every cycle
- E.g., 4 GHz Pentium 4
  - runs 4 B cycles/sec
  - 1 cycle = 0.25 ns = 250 ps

Iron law

- Time/program = instrs/program x cycles/instr x sec/cycle
- sec/cycle (a.k.a. cycle time, clock time) - 'heartbeat' of computer
  - mostly determined by technology and CPU organization
- cycles/instr (a.k.a. CPI)
  - mostly determined by ISA and CPU organization
  - overlap among instructions makes this smaller
- instr/program (a.k.a. instruction count)
  - instrs executed NOT static code
  - mostly determined by program, compiler, ISA

Our Goal

- Minimize time which is the product, NOT isolated terms
- Common error to miss terms while devising optimizations
  - E.g., ISA change to decrease instruction count
  - BUT leads to CPU organization which makes clock slower

Iron Law Example

- Machine A: clock 1 ns, CPI 2.0, for a program
- Machine B: clock 2 ns, CPI 1.2, for same program
- Thus, Machine A is 1 GHz while B is lowly 500 MHz

- Which is faster and by how much?
- Time/program = instrs/program x cycles/instr x sec/cycle
  - Time(A): N x 2.0 x 1 = 2N
  - Time(B): N x 1.2 x 2 = 2.4N

- Compare: Time(B)/Time(A) = 2.4N/2N = 1.2
- On this program, Machine A is 20% faster than B

Iron Law Example

- Keep clock of A at 1 ns and clock of B at 2 ns
- For equal performance, if CPI of B is 1.2, what is A's CPI?
  - Time(B)/Time(A) = 1 = (N x 2 x 1.2)/(N x 1 x CPI(A))
  - CPI(A) = 2.4

Iron Law Example

- Keep CPI of A 2.0 and CPI of B 1.2
- For equal performance, if clock of B is 2 ns, what is A's clock?
  - Time(B)/Time(A) = 1 = (N x 2.0 x clock(A))/(N x 1.2 x 2)
  - clock(A) = 1.2 ns

Beware of Millions of Instr / Sec

- MIPS = instruction count/(execution time x 10^6)
  = clock rate/(CPI x 10^6) (How?)
- Often ignores program & quotes "peak"
  - ideal conditions => guarantee not to exceed!!
- Ignores instruction/program changes
  - E.g., adding floating-point H/W can hurt MIPS
  - 50 simple instructions replace by one slow FP op
- Okay if
  - instrs/program constant (e.g. same executable)
  - real program; not peak

Beware of Millions of FP Ops / Sec

- MFLOPS = FP ops in program/(execution time x 10^6)
- Assumes FP ops independent of compiler/ISA
  - Assumption not true
  - may not have divide instruction in ISA
  - optimizing compilers can remove
- Relative MIPS and normalized MFLOPS
  - adds to confusion! (see book)

Rules

- Use ONLY Time
  - Beware when reading, especially if details are omitted
  - Beware of Peak

Which Programs?

- Execution time of what?
- Best case - you always run the same set of programs
  - port them and time the whole "workload"
- In reality, use benchmarks
  - programs chosen to measure performance
  - predict performance of actual workload (hopefully)
  - saves effort and money
  - representative? honest?
  - Example Suites: EEMBC, MediaBench, SPEC, &TPC

Benchmarks: SPEC CPU2000

- SPEC: System Performance Evaluation Cooperative
- Latest is SPEC2K, before SPEC89, SPEC92, SPEC95
- 12 integer and 14 floating point programs
  - GM of the normalized times

SPEC CPU2000 Integer

Benchmark	Description
gzip	Compression
vpr	FPGA place/route
gcc	GNU C compiler
mcf	Combinatorial optimizer
crafty	Chess
parser	Word processing
eon	Visualization
perlbmk	Perl application
gap	Group theory
vortex	Object-oriented database
bzip2	Compression
twolf	Place/route simulator

SPEC CPU2000 Floating Point

Benchmark	Description
wupwise	Quantum chomodynamics
swim	Shallow water model
mgrid	Multigrid solver of 3D grid
applu	Parabolic/elliptic PDEs
mesa	3D graphics library
galgel, art, equake, facerec, ammp, lucas, fma3d, sixtrack, apsi	Remaining 9 FP applications

SPECfp95

Benchmark	Description
su2cor	Monte Carlo
mgrid	3-D potential field
wave5	EM particle simulation
hydro2d	Navier Stokes Equations

How to Average

- Example

	Machine A (sec)	Machine B (sec)
Program 1	1	10
Program 2	1000	100
Total	1001	110

- One answer: total execution time
- Then B is how much faster than A? 9.1

How to Average

- Another: arithmetic mean (same result: B 9.1 times faster than A)
- Arithmetic mean of times $\left\{ \sum_{=} ne_i \right\} n$ for n programs
- AM(A) = 1001/2 = 500.5
- AM(B) = 110/2 = 55
- 500.5/55 = 9.1
- Valid only if programs run equally often, else use "weight" factors
- Weighted arithmetic mean: $\left\{ \sum_{=} ight_i \times ime_i \right\} n$

CS/ECE 552 (25) Sankaralingam

Other Averages

- E.g., 30 mph for first 10 miles
- 90 mph for next 10 miles. Average speed?
- Average speed = (30+90)/2 =60mph? WRONG
- Average speed = total distance / total time
  = (20 / (10/30+10/90))
  = 45 mph
- What if it was 10 hours at each speed?
  - instead of 10 miles

CS/ECE 552 (26) Sankaralingam

Harmonic Mean

- Harmonic mean of rates = $\dfrac{1}{\left\{ \sum_{=1} \dfrac{1}{ate_i} \right\} / n}$
  - Use HM if forced to start and end with rates
- Trick to do arithmetic mean of times but using rates and not times

CS/ECE 552 (27) Sankaralingam

Dealing with Ratios

- Absolute execution times (sec)

	Machine A	Machine B
Program 1	1	10
Program 2	1000	100

- Now consider ratios (w.r.t. A)

	Machine A	Machine B
Program 1	1	10
Program 2	1	0.1

- Averages: A = 1, B = 5.05

CS/ECE 552 (28) Sankaralingam

Dealing with Ratios

- Absolute execution times (sec)

	Machine A	Machine B
Program 1	1	10
Program 2	1000	100

- Now consider ratios (w.r.t. B)

	Machine A	Machine B
Program 1	0.1	1
Program 2	10	1

- Averages: A = 5.05, B = 1 Both cannot be true!

Geometric Mean

- Don't use arithmetic mean on ratios (normalized numbers)
- Use geometric mean for ratios
  - geometric mean of ratios =
  - Use GM if forced to use ratios $\sqrt[n]{\prod_{i=1} {}'io_i}$
- Independent of reference machine (math property)
- In the example, GM for machine A is 1, for machine B is also 1
- Normalized with respect to either machine
- Used in SPECint and SPECfp

But..

- Geometric mean of ratios is not proportional to total time
- AM in example says machine B is 9.1 times faster
- GM says they are equal
- If we took total execution time, A and B are equal only if
  - program 1 is run 100 times more often than program 2
- Generally, GM will mispredict for three or more machines

Summary for Averages

- Use AM for times
- Use HM if forced to use rates
- Use GM if forced to use ratios

- Better yet
  - Use unnormalized numbers to compute time

Amdahl's Law

- Why does the common case matter the most?
- Let an optimization speed f fraction of time by a factor of s
- assuming that old time = T, what is the speedup?
  - f is the "affected" fraction of T
  - (1-f) is the unaffected fraction

- Speedup =
- $$= \frac{time_{old}}{time_{new}} = \frac{unaffected_{old} + affected_{old}}{unaffected_{new} + affected_{new}}$$

$$\frac{(1-f)\times T + f\times T}{(1-f)\times T + \frac{f}{s}\times T}$$

Amdahl's Law Example

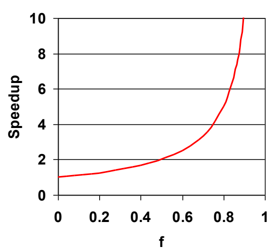
- Your boss asks you to improve processor performance
- Two options: What should you do?
  - improve the ALU used 95% of time, by 10%
  - improve the square-root unit used 5%, by a factor of 10

f	s	Speedup
95%	1.10	1.094
5%	10	1.047
5%	∞	1.052

Amdahl's Law: Limit

- Make common case fast because:

$$\lim_{s\to\infty}\left(\frac{1}{1-f+f/s}\right) = \frac{1}{1-f}$$



Amdahl's Law

- "Make common case fast"
  - Heuristic, not commandment
  - Use for intuition, verify with numbers
- 60% can be improved by a factor of 2
  - Speedup = 1/(0.4+0.6/2) = 1/0.7
- 40% can be improved by a factor of 8
  - Speedup = 1/(0.6+0.4/8) = 1/0.65
- Second option is better
  - Less common case, but higher speedup compensates

Summary

- Time and performance:
  - Machine A n times faster than Machine B
  - iff Time(B)/Time(A) = n
- Iron Law: Time/prog
  - Instr count x CPI x Cycle time
- Other Metrics: MIPS and MFLOPS
  - Beware of peak and omitted details
- Benchmarks: SPEC95
- Summarize performance:
  - AM for time, HM for rate, GM for ratio
- Amdahl's Law: Speedup = $\left(\frac{1}{1-f+f/s}\right)$ common case fast