Computer System Performance

Virendra Singh

Associate Professor

Computer Architecture and Dependable Systems Lab

Department of Electrical Engineering

Indian Institute of Technology Bombay

http://www.ee.iitb.ac.in/~viren/

E-mail: viren@ee.iitb.ac.in

CP-226: Computer Architecture



Lecture 4 (01 Feb 2013)

CADSI

Performance and Cost

 Which of the following airplanes has the best performance?

Airplane	Passengers	Range (mi)	Speed (mph)
Boeing 737-100	101	630	598
Boeing 747	470	4150	610
BAC/Sud Concord	de 132	4000	1350
Douglas DC-8-50	146	8720	544

- How much faster is the Concorde vs. the 747
- How much bigger is the 747 vs. DC-8?



Performance and Cost

- Which computer is fastest?
- Not so simple
 - Scientific simulation FP performance
 - Program development Integer performance
 - Database workload Memory, I/O





Performance of Computers

- Want to buy the fastest computer for what you want to do?
 - Workload is all-important
 - Correct measurement and analysis
- Want to design the fastest computer for what the customer wants to pay?
 - Cost is an important criterion





Defining Performance

- What is important to whom?
- Computer system user
 - Minimize elapsed time for program = time_end time_start
 - Called response time
- Computer center manager
 - Maximize completion rate = #jobs/second
 - Called throughput





Response Time vs. Throughput

- Is throughput = 1/av. response time?
 - Only if NO overlap
 - Otherwise, throughput > 1/av. response time
 - E.g. a lunch buffet assume 5 entrees
 - Each person takes 2 minutes/entrée
 - Throughput is 1 person every 2 minutes
 - BUT time to fill up tray is 10 minutes
 - Why and what would the throughput be otherwise?
 - 5 people simultaneously filling tray (overlap)
 - Without overlap, throughput = 1/10





What is Performance for us?

- For computer architects
 - CPU time = time spent running a program
- Intuitively, bigger should be faster, so:
 - Performance = 1/X time, where X is response, CPU execution, etc.
- Elapsed time = CPU time + I/O wait
- We will concentrate on CPU time



Improve Performance

- Improve (a) response time or (b) throughput?
 - Faster CPU
 - Helps both (a) and (b)
 - Add more CPUs
 - Helps (b) and perhaps (a) 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. time(A) = 10s, time(B) = 15s
 - 15/10 = 1.5 => A is 1.5 times faster than B
 - 15/10 = 1.5 => 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
- For example:
 - 1GHz Snapdragon runs 1000M cycles/sec, 1 cycle = 1ns
 - 2.5GHz Core i7 runs 2.5G cycles/sec, 1 cycle = 0.25ns



Iron Law

$$= \frac{\text{Instructions}}{\text{Program}} \quad X \quad \frac{\text{Cycles}}{\text{Instruction}} \quad X \quad \frac{\text{Time}}{\text{Cycle}}$$
(code size) (CPI) (cycle time)

Architecture --> Implementation --> Realization

Compiler Designer Processor Designer Chip Designer



Iron Law

- Instructions/Program
 - Instructions executed, not static code size
 - Determined by algorithm, compiler, ISA
- Cycles/Instruction
 - Determined by ISA and CPU organization
 - Overlap among instructions reduces this term
- Time/cycle
 - Determined by technology, organization, clever circuit design



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
- Bottom line: terms are inter-related



Other Metrics

- MIPS and MFLOPS
- MIPS = instruction count/(execution time x 10⁶)
 - = clock rate/(CPI x 10^6)
- But MIPS has serious shortcomings

Problems with MIPS

- E.g. without FP hardware, an FP op may take 50 single-cycle instructions
- With FP hardware, only one 2-cycle instruction

Thus, adding FP hardware:

- CPI increases (why?)
- Instructions/program decreases (why?)
- Total execution time decreases
- BUT, MIPS gets worse!



Problems with MIPS

- Ignores program
- Usually used to quote peak performance
 - Ideal conditions => guaranteed not to exceed!
- When is MIPS ok?
 - Same compiler, same ISA
 - E.g. same binary running on AMD Phenom, Intel
 Core i7
 - Why? Instr/program is constant and can be ignored





Other Metrics

- MFLOPS = FP ops in program/(execution time x 10⁶)
- Assuming FP ops independent of compiler and ISA
 - Often safe for numeric codes: matrix size determines # of FP ops/program
 - However, not always safe:
 - Missing instructions (e.g. FP divide)
 - Optimizing compilers
- Relative MIPS and normalized MFLOPS
 - Adds to confusion



Rules

- Use ONLY Time
- Beware when reading, especially if details are omitted
- Beware of Peak
 - "Guaranteed not to exceed"



Iron Law Example

- Machine A: clock 1ns, CPI 2.0, for program x
- Machine B: clock 2ns, CPI 1.2, for program x
- Which is faster and how much?

Time/Program = instr/program x cycles/instr x sec/cycle

 $Time(A) = N \times 2.0 \times 1 = 2N$

 $Time(B) = N \times 1.2 \times 2 = 2.4N$

Compare: Time(B)/Time(A) = 2.4N/2N = 1.2

So, Machine A is 20% faster than Machine B for this program



Iron Law Example

Keep clock(A) @ 1ns and clock(B) @2ns
For equal performance, if CPI(B)=1.2, what is CPI(A)?

Time(B)/Time(A) = 1 = (Nx2x1.2)/(Nx1xCPI(A))CPI(A) = 2.4





Iron Law Example

- Keep CPI(A)=2.0 and CPI(B)=1.2
- For equal performance, if clock(B)=2ns, what is clock(A)?

Time(B)/Time(A) = 1 = $(N \times 2.0 \times clock(A))/(N \times 1.2 \times 2)$ clock(A) = 1.2ns



Which Programs

- Execution time of what program?
- Best case your 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
 - Saves effort and money
 - Representative? Honest? Benchmarketing...





How to Average

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

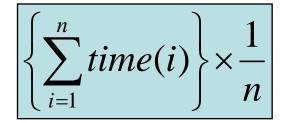
 One answer: for total execution time, how much faster is B? 9.1x





How to Average

- Another: arithmetic mean (same result)
- Arithmetic mean of times:
- AM(A) = 1001/2 = 500.5
- AM(B) = 110/2 = 55
- 500.5/55 = 9.1x



 Valid only if programs run equally often, so use weighted arithmetic mean:

$$\left\{ \sum_{i=1}^{n} \left(weight(i) \times time(i) \right) \right\} \times \frac{1}{n}$$



Other Averages

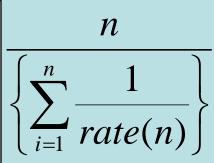
- E.g., 30 mph for first 10 miles, then 90 mph for next 10 miles, what is average speed?
- Average speed = (30+90)/2 WRONG
- Average speed = total distance / total time
 - = (20 / (10/30 + 10/90))
 - = 45 mph





Harmonic Mean

• Harmonic mean of rates =



- Use HM if forced to start and end with rates (e.g. reporting MIPS or MFLOPS)
- Why?
 - Rate has time in denominator
 - Mean should be proportional to inverse of sums of time (not sum of inverses)
 - See: J.E. Smith, "Characterizing computer performance with a single number," CACM Volume 31, Issue 10 (October 1988), pp. 1202-1206.



Dealing with Ratios

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

If we take ratios with respect to machine A

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



Dealing with Ratios

- Average for machine A is 1, average for machine B is 5.05
- If we take ratios with respect to machine B

	Machine A	Machine B
Program 1	0.1	1
Program 2	10	1
Average	5.05	1

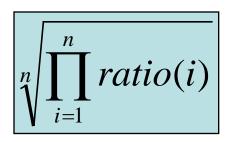
- Can't both be true!!!
- Don't use arithmetic mean on ratios!





Geometric Mean

- Use geometric mean for ratios
- Geometric mean of ratios =



- Independent of reference machine
- In the example, GM for machine a is 1, for machine B is also 1
 - Normalized with respect to either machine



But...

- GM 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

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

 Best of all, use unnormalized numbers to compute time

Benchmarks: SPEC2000

- System Performance Evaluation Cooperative
 - Formed in 80s to combat benchmarketing
 - SPEC89, SPEC92, SPEC95, SPEC2000
- 12 integer and 14 floating-point programs
 - Sun Ultra-5 300MHz reference machine has score of 100
 - Report GM of ratios to reference machine





Benchmarks: SPEC CINT2000

Benchmark	Description
164.gzip	Compression
175.vpr	FPGA place and route
176.gcc	C compiler
181.mcf	Combinatorial optimization
186.crafty	Chess
197.parser	Word processing, grammatical analysis
252.eon	Visualization (ray tracing)
253.perlbmk	PERL script execution
254.gap	Group theory interpreter
255.vortex	Object-oriented database
256.bzip2	Compression
300.twolf	Place and route simulator



Benchmarks: SPEC CFP2000

Benchmark	Description
168.wupwise	Physics/Quantum Chromodynamics
171.swim	Shallow water modeling
172.mgrid	Multi-grid solver: 3D potential field
173.applu	Parabolic/elliptic PDE
177.mesa	3-D graphics library
178.galgel	Computational Fluid Dynamics
179.art	Image Recognition/Neural Networks
183.equake	Seismic Wave Propagation Simulation
187.facerec	Image processing: face recognition
188.ammp	Computational chemistry
189.lucas	Number theory/primality testing
191.fma3d	Finite-element Crash Simulation
200.sixtrack	High energy nuclear physics accelerator design
301.apsi	Meteorology: Pollutant distribution



Thank You



