inst.eecs.berkeley.edu/~cs61c

CS61C : Machine Structures

Lecture 41
Performance II

Lecturer PSOE Dan Garcia

www.cs.berkeley.edu/~ddgarcia

UWB…Ultra Wide Band! ⇒ The FCC moved one step closer to approving a standard for this technology which uses spread spectrum pulses to send its information. Imagine no data wires to ANY of your devices!

www.nytimes.com/2005/05/04/technology/techspecial/04markoff.html

What is Time?

- Straightforward definition of time:
  - Total time to complete a task, including disk accesses, memory accesses, I/O activities, operating system overhead, ...
  - "real time", "response time" or "elapsed time"
- Alternative: just time processor (CPU) is working only on your program (since multiple processes running at same time)
  - "CPU execution time" or "CPU time"
  - Often divided into system CPU time (in OS) and user CPU time (in user program)

How to Measure Time?

- User Time ⇒ seconds
- CPU Time: Computers constructed using a clock that runs at a constant rate and determines when events take place in the hardware
  - These discrete time intervals called clock cycles (or informally clocks or cycles)
  - Length of clock period: clock cycle time (e.g., 2 nanoseconds or 2 ns) and clock rate (e.g., 500 megahertz, or 500 MHz), which is the inverse of the clock period; use these!

Measuring Time using Clock Cycles (1/2)

- CPU execution time for a program

= Clock Cycles for a program x Clock Cycle Time

- or

$$= \frac{\text{Clock Cycles for a program}}{\text{Clock Rate}}$$

Measuring Time using Clock Cycles (2/2)

- One way to define clock cycles:

Clock Cycles for program

= Instructions for a program (called "Instruction Count")

x Average Clock cycles Per Instruction (abbreviated "CPI")

- CPI one way to compare two machines with same instruction set, since Instruction Count would be the same

Performance Calculation (1/2)

- CPU execution time for program
  = Clock Cycles for program x Clock Cycle Time
- Substituting for clock cycles:

CPU execution time for program = (Instruction Count x CPI) x Clock Cycle Time

= Instruction Count x CPI x Clock Cycle Time

Performance Calculation (2/2)

$$\text{CPU time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}}$$

- Product of all 3 terms: if missing a term, can't predict time, the real measure of performance

How Calculate the 3 Components?

- Clock Cycle Time: in specification of computer (Clock Rate in advertisements)
- Instruction Count:
  - Count instructions in loop of small program
  - Use simulator to count instructions
  - Hardware counter in spec. register
    - (Pentium II,III,4)
- CPI:
  - Calculate: $\frac{\text{Execution Time / Clock cycle time}}{\text{Instruction Count}}$
  - Hardware counter in special register (PII,III,4)

Calculating CPI Another Way

- First calculate CPI for each individual instruction (`add`, `sub`, `and`, etc.)
- Next calculate frequency of each individual instruction
- Finally multiply these two for each instruction and add them up to get final CPI (the weighted sum)

Example (RISC processor)

Op	$Freq_i$	CPI_i	Prod	(% Time)
ALU	50%	1	.5	(23%)
Load	20%	5	1.0	(45%)
Store	10%	3	.3	(14%)
Branch	20%	2	.4	(18%)
			2.2	

Instruction Mix (Where time spent)

- What if Branch instructions twice as fast?

What Programs Measure for Comparison?

- Ideally run typical programs with typical input before purchase, or before even build machine
  - Called a "workload"; For example:
  - Engineer uses compiler, spreadsheet
  - Author uses word processor, drawing program, compression software
- In some situations its hard to do
  - Don't have access to machine to "benchmark" before purchase
  - Don't know workload in future
- Next: benchmarks & PC-Mac showdown!

Benchmarks

- Obviously, apparent speed of processor depends on code used to test it
- Need industry standards so that different processors can be fairly compared
- Companies exist that create these benchmarks: "typical" code used to evaluate systems
- Need to be changed every 2 or 3 years since designers could (and do!) target for these standard benchmarks

Example Standardized Benchmarks (1/2)

- Standard Performance Evaluation Corporation (SPEC) SPEC CPU2000
  - CINT2000 12 integer (gzip, gcc, crafty, perl, ...)
  - CFP2000 14 floating-point (swim, mesa, art, ...)
  - All relative to base machine Sun 300MHz 256Mb-RAM Ultra5_10, which gets score of 100
  - `www.spec.org/osg/cpu2000/`
  - They measure
    - System speed (SPECint2000)
    - System throughput (SPECint_rate2000)

Example Standardized Benchmarks (2/2)

- SPEC
  - Benchmarks distributed in source code
  - Members of consortium select workload
    - 30+ companies, 40+ universities
  - Compiler, machine designers target benchmarks, so try to change every 3 years
  - The last benchmark released was SPEC 2000
    - They are still finalizing SPEC 2005

CINT2000

gzip	C	Compression
vpr	C	FPGA Circuit Placement and Routing
gcc	C	C Programming Language Compiler
mcf	C	Combinatorial Optimization
crafty	C	Game Playing: Chess
parser	C	Word Processing
eon	C++	Computer Visualization
perlbmk	C	PERL Programming Language
gap	C	Group Theory, Interpreter
vortex	C	Object-oriented Database
bzip2	C	Compression
twolf	C	Place and Route Simulator

CFP2000

wupwise	Fortran77	Physics / Quantum Chromodynamics
swim	Fortran77	Shallow Water Modeling
mgrid	Fortran77	Multi-grid Solver: 3D Potential Field
applu	Fortran77	Parabolic / Elliptic Partial Differential Equations
mesa	C	3-D Graphics Library
galgel	Fortran90	Computational Fluid Dynamics
art	C	Image Recognition / Neural Networks
equake	C	Seismic Wave Propagation Simulation
facerec	Fortran90	Image Processing: Face Recognition
ammp	C	Computational Chemistry
lucas	Fortran90	Number Theory / Primality Testing
fma3d	Fortran90	Finite-element Crash Simulation
sixtrack	Fortran77	High Energy Nuclear Physics Accelerator Design
apsi	Fortran77	Meteorology: Pollutant Distribution

Example PC Workload Benchmark

- PCs: Ziff-Davis Benchmark Suite
  - "Business Winstone is a system-level, application-based benchmark that measures a PC's overall performance when running today's top-selling Windows-based 32-bit applications… it doesn't mimic what these packages do; it runs real applications through a series of scripted activities and uses the time a PC takes to complete those activities to produce its performance scores.
  - Also tests for CDs, Content-creation, Audio, 3D graphics, battery life

`http://www.etestinglabs.com/benchmarks/`

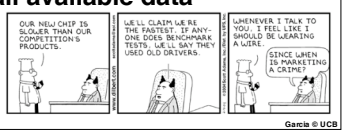
Performance Evaluation

- Good products created when have:
  - Good benchmarks
  - Good ways to summarize performance
- Given sales is a function of performance relative to competition, should invest in improving product as reported by performance summary?
- If benchmarks/summary inadequate, then choose between improving product for real programs vs. improving product to get more sales; Sales almost always wins!

Performance Evaluation: The Demo

If we're talking about performance, let's discuss the ways shady salespeople have fooled consumers (so that you don't get taken!)

5. Never let the user touch it
4. Only run the demo through a script
3. Run it on a stock machine in which "no expense was spared"
2. Preprocess all available data
1. Play a movie

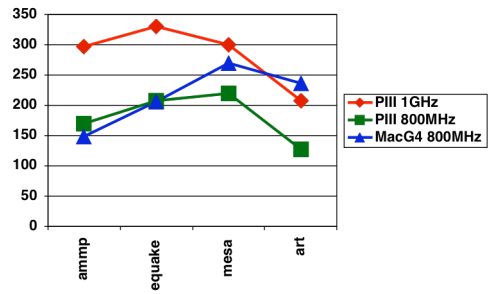


PC / PC / Mac Showdown!!! (1/4)

- PC
  - 1 GHz Pentium III
  - 256 Mb RAM
  - 512KB L2 Cache
  - No L3
  - 133 MHz Bus
  - 20 GB Disk
  - 16MB VRAM
- PC 800MHz PIII
- Mac
  - 800 MHz PowerbookG4
  - 1 Gb RAM
    - 2 512Mb SODIMMs
  - 32KB L1Inst, L1Data
  - 256KB L2 Cache
  - 1Mb L3 Cache
  - 133 MHz Bus
  - 40 GB Disk
  - 32MB VRAM

Let's take a look at SPEC2000 and a simulation of a real-world application.

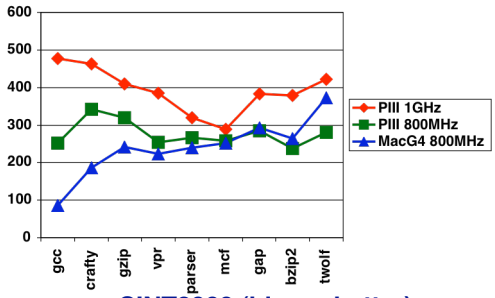
PC / Mac Showdown!!! (2/4)



CFP2000 (bigger better)

[left-to-right by G4/PIII 800MHz ratio]

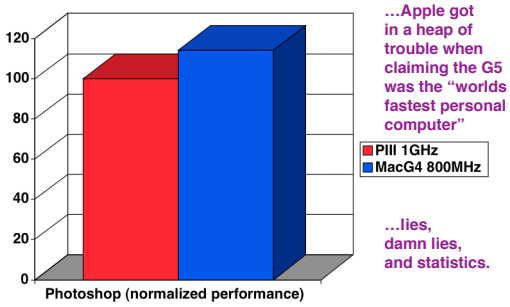
PC / Mac Showdown!!! (3/4)



CINT2000 (bigger better)

[left-to-right by G4/PIII 800MHz ratio]

PC / Mac Showdown!!! (4/4)



…Apple got in a heap of trouble when claiming the G5 was the "worlds fastest personal computer"

…lies, damn lies, and statistics.

Normalized Photoshop radial blur (bigger better)

[Amt=10,Zoom,Best](PIII = 79sec = "100", G4= 69sec)

Administrivia

- HKN evaluations on Monday
- Final survey in lab this week
- Last semester's final + solutions online
- Final exam review
  - Sunday, 2005-05-08 @ 2pm in 10 Evans
- Final exam
  - Tuesday, 2005-05-14 @ 12:30-3:30pm in 220 Hearst Gym
  - Same rules as Midterm, except you get 2 double-sided handwritten review sheets (1 from your midterm, 1 new one) + green sheet [Don't bring backpacks]

Peer Instruction

A. Performance is a stinking business; easily corruptible and (in general) you won't hear honest reports from a company if they have a vested interest in the results.

B. The Sieve of Eratosthenes, Puzzle and Quicksort were early effective benchmarks.

C. A program runs in 100 sec. on a machine, `mult` accounts for 80 sec. of that. If we want to make the program run 6 times faster, we need to up the speed of `mult`s by AT LEAST 6.

	ABC
1:	FFF
2:	FFT
3:	FTF
4:	FTT
5:	TFF
6:	TFT
7:	TTF
8:	TTT

"And in conclusion…"

$$\text{CPU time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

- Latency v. Throughput
- Performance doesn't depend on any single factor: need Instruction Count, Clocks Per Instruction (CPI) and Clock Rate to get valid estimations
- User Time: time user waits for program to execute: depends heavily on how OS switches between tasks
- CPU Time: time spent executing a single program: depends solely on design of processor (datapath, pipelining effectiveness, caches, etc.)
- Benchmarks
  - Attempt to predict perf, Updated every few years
  - Measure everything from simulation of desktop graphics programs to battery life
- Megahertz Myth
  - MHz ≠ performance, it's just one factor