inst.eecs.berkeley.edu/~cs61c

CS61C : Machine Structures

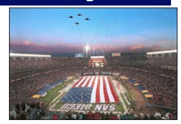
Lecture 41
Performance I

2004-12-06

Lecturer PSOE Dan Garcia

www.cs.berkeley.edu/~ddgarcia

Sour Roses! ⇒



Cal's best season in two generations (since 1949) deserved a Rose Bown berth! It's a mockery of a sham of a travesty of 2 shams.

Holiday Bowl?! vs #23 TexasTech?

calbears.collegesports.com/sports/m-footbl/spec-rel/120504aaa.html

Review

- Magnetic disks continue rapid advance: 2x/yr capacity, 2x/2-yr bandwidth, slow on seek, rotation improvements, MB/$ 2x/yr!
  - Designs to fit high volume form factor
- RAID
  - Higher performance with more disk arms per $
  - Adds option for small # of extra disks (the "R")
  - Started @ Cal by CS Profs Katz & Patterson

Cool addition to the last lecture

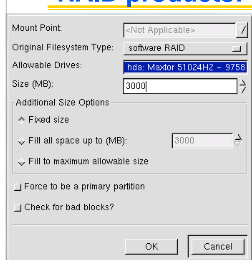
- Drives inside the iPod and iPod Mini:

Thanks to Andy Dahl for the tip

Hitachi 1 inch 4GB MicroDrive

Toshiba 1.8-inch 20/40/60GB (MK1504GAL)

RAID products: Software, Chips, Systems



RAID is $32 B industry in 2002, 80% nonPC disks sold in RAIDs

Margin of Safety in CS&E?

- Patterson reflects…
  - Operator removing good disk vs. bad disk
  - Temperature, vibration causing failure before repair
  - In retrospect, suggested RAID 5 for what we anticipated, but should have suggested RAID 6 (double failure OK) for unanticipated/safety margin…

Administrivia

- Dan releases last semester's final + answers online soon
- HKN evaluations on Friday
- Final survey in lab this week
- Final exam review
  - Sunday, 2004-12-12 @ 2pm in 10 Evans
- Final exam
  - Tuesday, 2004-12-14 @ 12:30-3:30pm in 230 Hearst Gym
  - Same rules as Midterm, except you get 2 double-sided handwritten review sheets (1 from your midterm, 1 new one) + green sheet

Performance

- **Purchasing Perspective**: given a collection of machines (or upgrade options), which has the
  - best performance ?
  - least cost ?
  - best performance / cost ?
- **Computer Designer Perspective**: faced with design options, which has the
  - best performance improvement ?
  - least cost ?
  - best performance / cost ?
- All require basis for comparison and metric for evaluation
- Solid metrics lead to solid progress!

Two Notions of "Performance"

Plane	DC to Paris	Top Speed	Passen-gers	Throughput (pmph)
Boeing 747	6.5 hours	610 mph	470	286,700
BAD/Sud Concorde	3 hours	1350 mph	132	178,200

- Which has higher performance?
  - Time to deliver 1 passenger?
  - Time to deliver 400 passengers?
- In a computer, time for 1 job called
  - Response Time or Execution Time
- In a computer, jobs per day called
  - Throughput or Bandwidth

Definitions

- Performance is in units of things per sec
  - bigger is better
- If we are primarily concerned with response time
  - $\text{performance}(x) = \dfrac{1}{\text{execution\_time}(x)}$

" F(ast) is *n* times faster than S(low) " means…

$$n = \frac{\text{performance}(F)}{\text{performance}(S)} = \frac{\text{execution\_time}(S)}{\text{execution\_time}(F)}$$

Example of Response Time v. Throughput

- Time of Concorde vs. Boeing 747?
  - Concord is 6.5 hours / 3 hours = 2.2 times faster
- Throughput of Boeing vs. Concorde?
  - Boeing 747: 286,700 pmph / 178,200 pmph = 1.6 times faster
- Boeing is 1.6 times ("60%") faster in terms of throughput
- Concord is 2.2 times ("120%") faster in terms of flying time (response time)

We will focus primarily on execution time for a single job

Confusing Wording on Performance

- Will (try to) stick to "n times faster"; its less confusing than "m % faster"
- As faster means both increased performance and decreased execution time, to reduce confusion we will (and you should) use "improve performance" or "improve execution time"

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)

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

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

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

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

- 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%)
	Instruction Mix		2.2	(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
- Wednesday: 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

Peer Instruction

A. Clock rate does not equal performance.

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…"

- Latency v. Throughput
- Performance doesn't depend on any single factor: need to know Instruction Count, Clocks Per Instruction (CPI) and Clock Rate to get valid estimations
- User Time: time user needs to wait 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.)

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