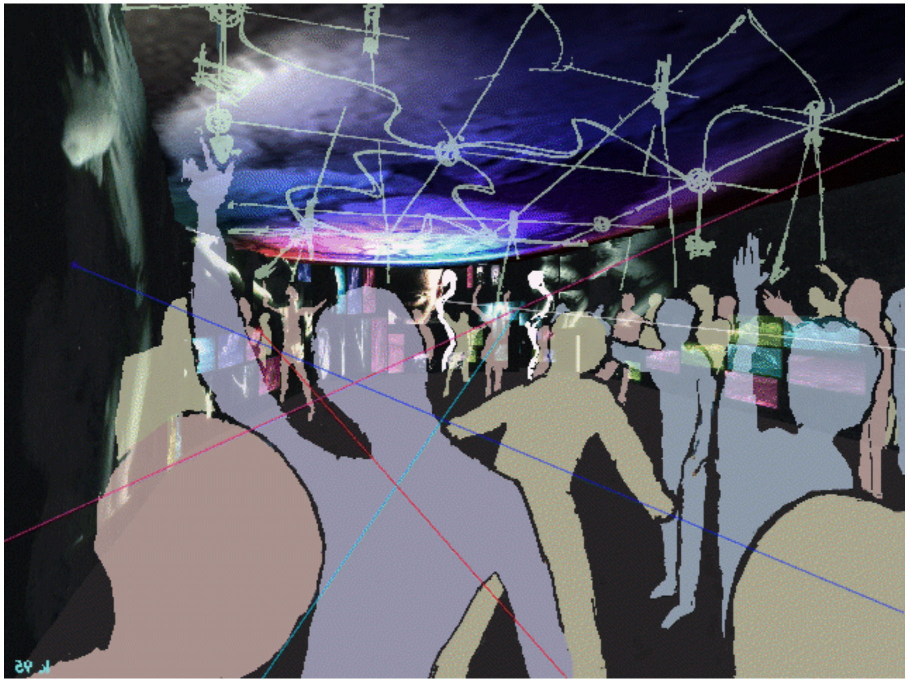
inst.eecs.berkeley.edu/~cs61c/su05

CS61C : Machine Structures

Lecture #27: RAID & Performance



2005-08-08

Andy Carle

Outline

- RAID
- Performance

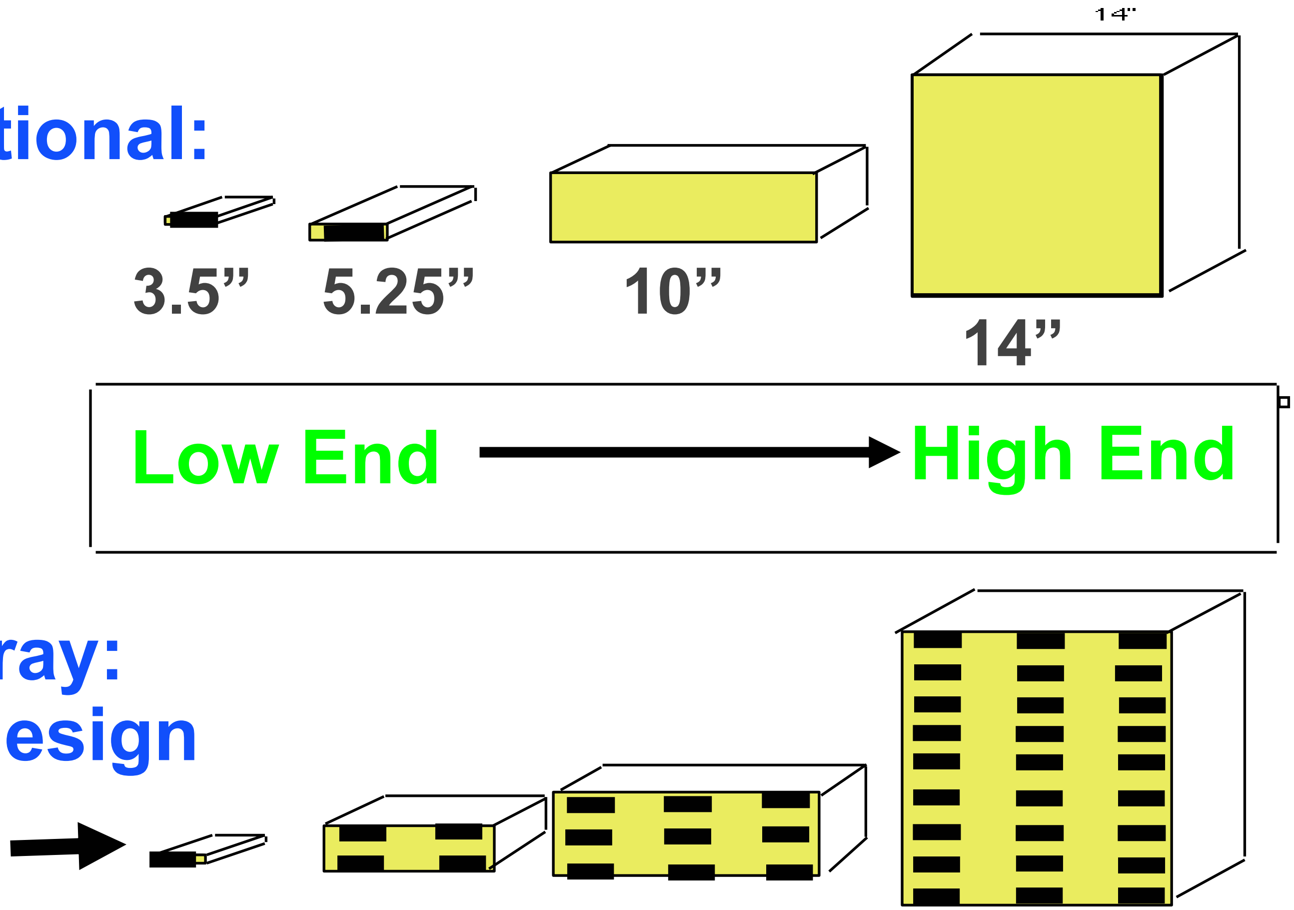
Use Arrays of Small Disks…

- **Katz and Patterson asked in 1987:**
  - **Can smaller disks be used to close gap in performance between disks and CPUs?**

Conventional: 4 disk designs

Disk Array: 1 disk design

3.5”



Replace Small Number of Large Disks with Large Number of Small Disks! (1988 Disks)

	IBM 3390K	IBM 3.5" 0061	x70	
Capacity	20 GBytes	320 MBytes	23 GBytes	
Volume	97 cu. ft.	0.1 cu. ft.	11 cu. ft.	9X
Power	3 KW	11 W	1 KW	3X
Data Rate	15 MB/s	1.5 MB/s	120 MB/s	8X
I/O Rate	600 I/Os/s	55 I/Os/s	3900 IOs/s	6X
MTTF	250 KHrs	50 KHrs	??? Hrs	
Cost	$250K	$2K	$150K	

Disk Arrays potentially high performance, high MB per cu. ft., high MB per KW,

but what about reliability?

Array Reliability

- **Reliability** - whether or not a component has failed
  - measured as Mean Time To Failure (MTTF)
  - Reliability of N disks
    = Reliability of 1 Disk ÷ N
    (assuming failures independent)
    - 50,000 Hours ÷ 70 disks = 700 hour
  - Disk system MTTF:
    Drops from 6 years to 1 month!
  - Disk arrays (JBOD) too unreliable to be useful!

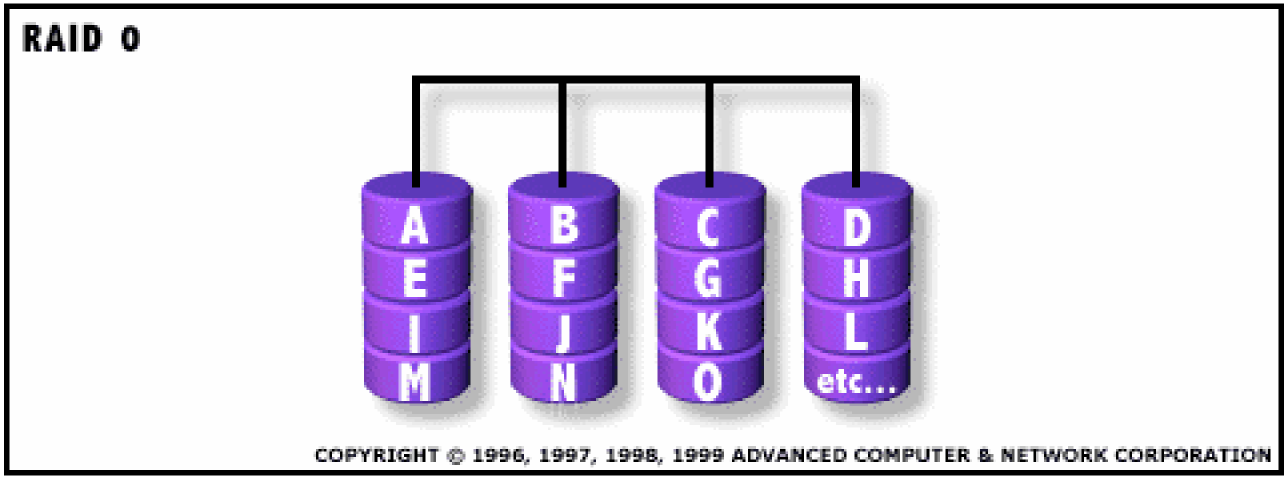
Redundant Arrays of (Inexpensive) Disks

- Files are "striped" across multiple disks
- Redundancy yields high data availability
  - **Availability**: service still provided to user, even if some components failed
- Disks will still fail
- Contents reconstructed from data redundantly stored in the array
  - ⇒ Capacity penalty to store redundant info
  - ⇒ Bandwidth penalty to update redundant info

Berkeley History, RAID-I

- RAID-I (1989)
  - Consisted of a Sun 4/280 workstation with 128 MB of DRAM, four dual-string SCSI controllers, 28 5.25-inch SCSI disks and specialized disk striping software
- Today RAID is $27 billion dollar industry, 80% nonPC disks sold in RAIDs

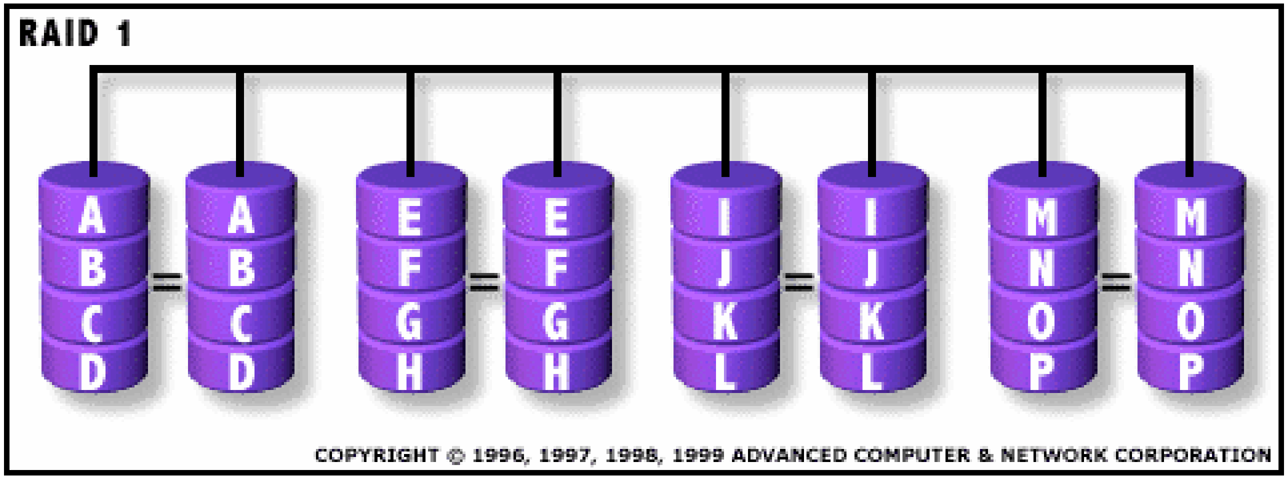
"RAID 0": Striping



- Assume have 4 disks of data for this example, organized in blocks
- Large accesses faster since transfer from several disks at once

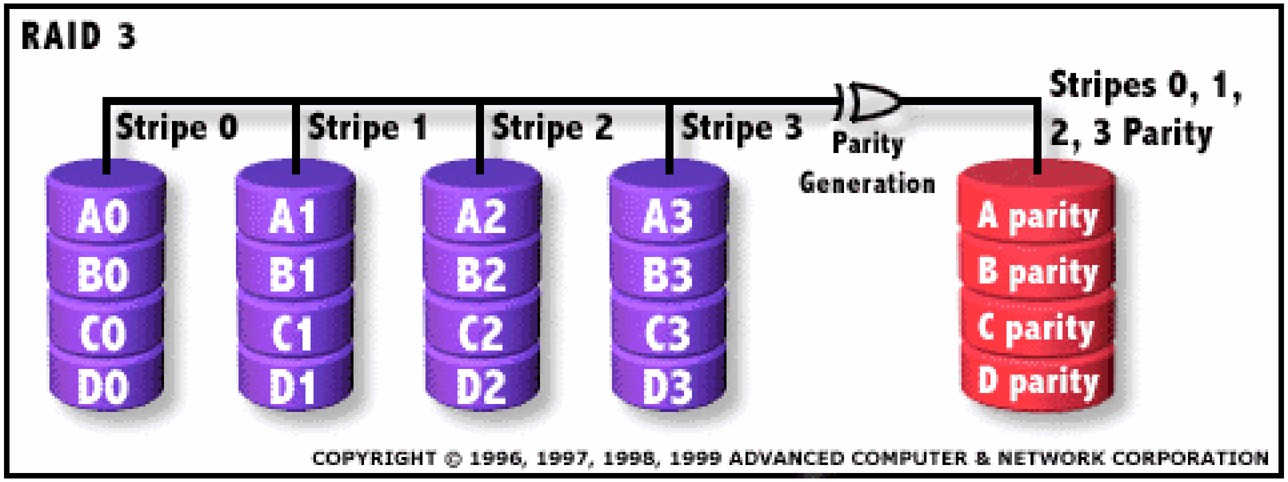
This and next 5 slides from RAID.edu, http://www.acnc.com/04_01_00.html

RAID 1: Mirror



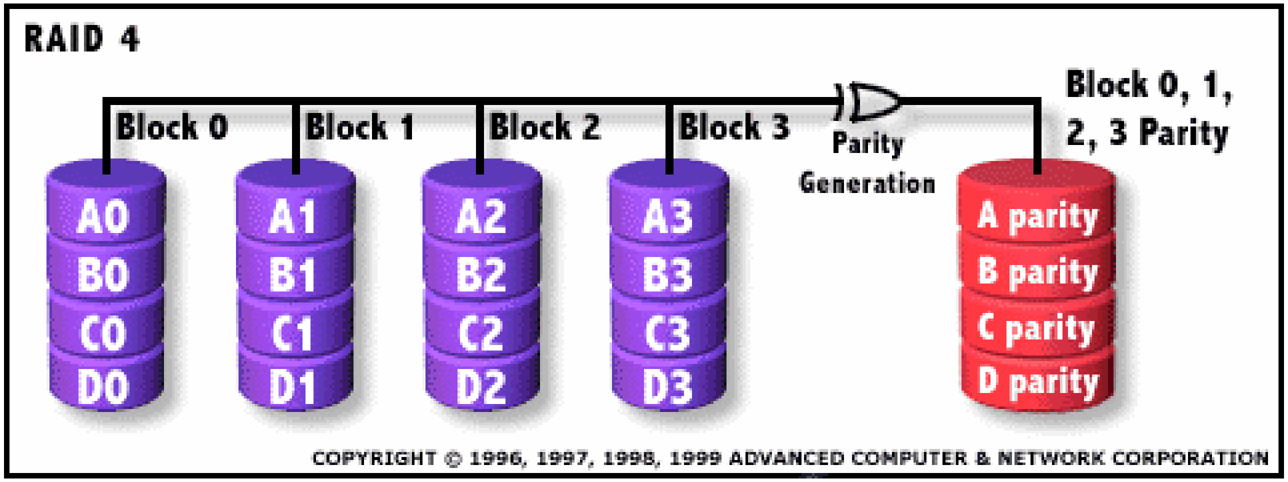
- Each disk is fully duplicated onto its "mirror"
  - Very high availability can be achieved
- Bandwidth reduced on write:
  - 1 Logical write = 2 physical writes
- Most expensive solution: 100% capacity overhead

RAID 3: Parity



- Parity computed across group to protect against hard disk failures, stored in P disk
- Logically, a single high capacity, high transfer rate disk
- 25% capacity cost for parity in this example vs. 100% for RAID 1 (5 disks vs. 8 disks)

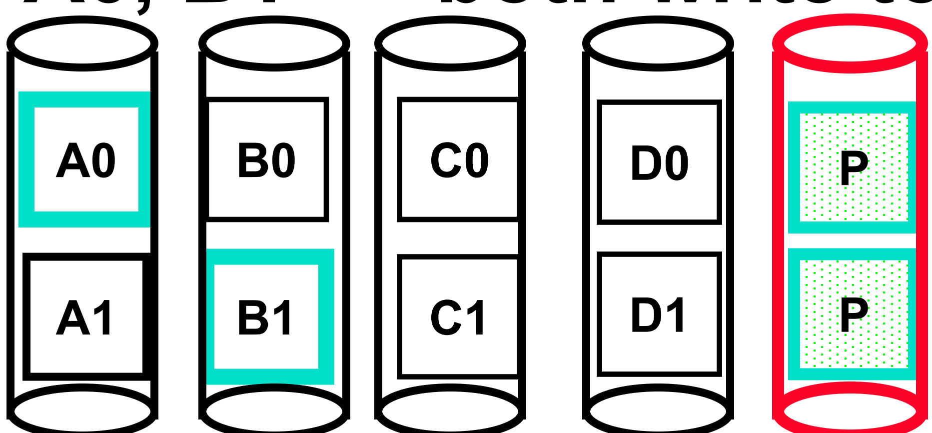
RAID 4: parity plus small sized accesses



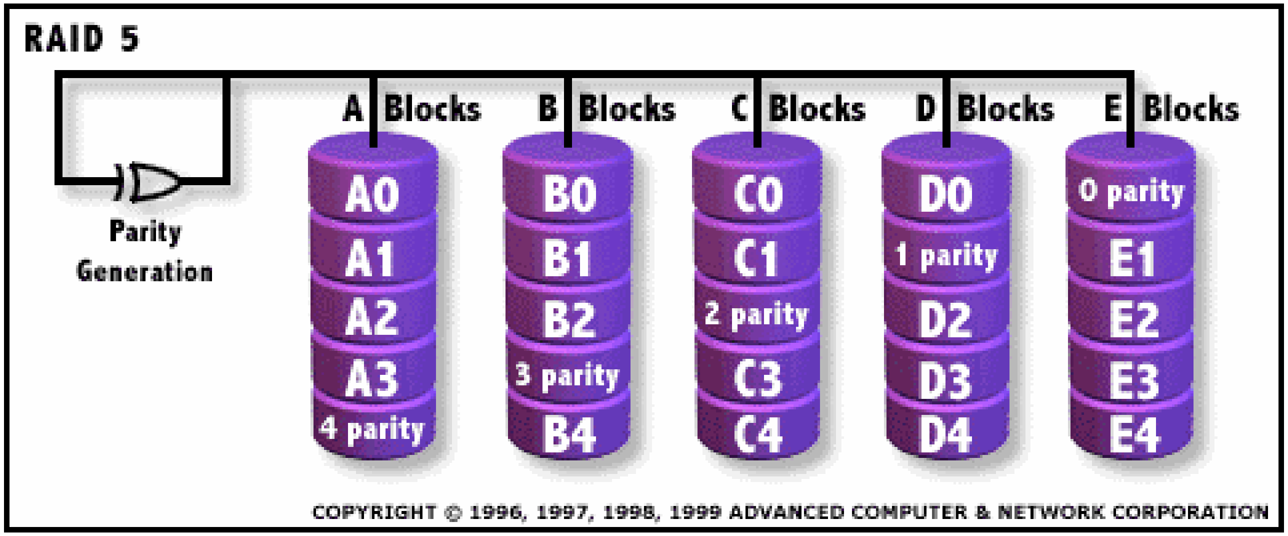
- RAID 3 relies on parity disk to discover errors on Read
- But every sector has an error detection field
- Rely on error detection field to catch errors on read, not on the parity disk
- Allows small independent reads to different disks simultaneously

Inspiration for RAID 5

- Small writes (write to one disk):
  - Option 1: read other data disks, create new sum and write to Parity Disk (access all disks)
  - Option 2: since P has old sum, compare old data to new data, add the difference to P: 1 logical write = 2 physical reads + 2 physical writes to 2 disks
- Parity Disk is bottleneck for Small writes: Write to A0, B1 => both write to P disk



RAID 5: Rotated Parity, faster small writes



- Independent writes possible because of interleaved parity
  - Example: write to A0, B1 uses disks 0, 1, 4, 5, so can proceed in parallel
  - Still 1 small write = 4 physical disk accesses

Outline

- RAID
- Performance

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

Administrivia

- Final Exam:
  - Friday, August 12, 11:00 – 2:00
  - 306 Soda (Same as Midterm 2)
  - Same rules as Midterms, except you can now have a two-sided cheat sheet
- Project 4: Due Friday
- HW8: Due Friday, but…
  - It is optional
    - The grade will be dropped if it hurts your overall semester grade
  - You may want to review it before the final

Upcoming Schedule

- Tuesday
  - Parallel Computing
  - HKN Evaluations (please BE HERE!)
  - Course Survey in lab
- Wednesday
  - Intro to Intel Architecture
  - Mini Review session in the remaining time
- Thursday
  - Official Review Session
- Friday: Final!

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", "elapsed time" or "wall 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 $\Rightarrow$ 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 program

$$= \text{Clock Cycles for a program} \times \text{Clock Cycle Time}$$

- 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)

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

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

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

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

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

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?

Example: What about Caches?

- Can Calculate Memory portion of CPI separately
- Miss rates: say L1 cache = 5%, L2 cache = 10%
- Miss penalties: L1 = 5 clock cycles, L2 = 50 clocks
- Assume miss rates, miss penalties same for instruction accesses, loads, and stores
- CPI_{memory}
  = Instruction Frequency * L1 Miss rate *
(L2 hit time + L2 miss rate * L2 miss penalty)
+ Data Access Frequency * L1 Miss rate *
(L2 hit time + L2 miss rate * L2 miss penalty)
  = 100%*5%*(5+10%*50)+(20%+10%)*5%*(5+10%*50)
  = 5%*(10)+(30%)*5%*(10) = 0.5 + 0.15 = 0.65

Overall CPI = 2.2 + 0.65 = 2.85

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 it's hard to do
  - Don't have access to machine to "benchmark" before purchase
  - Don't know workload in future

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
  - Big Company representatives select workload
    - Sun, HP, IBM, etc.
  - Compiler, machine designers target benchmarks, so try to change every 3 years

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

Performance Summary

- Benchmarks
  - Attempt to predict performance
  - Updated every few years
  - Measure everything from simulation of desktop graphics programs to battery life
- Megahertz Myth
  - MHz ≠ performance, it's just one factor
- It's non-trivial to try to help people in developing countries with technology
- Viruses have damaging potential the likes of which we can only imagine.

Megahertz Myth Marketing Video

http://a256.g.akamai.net/5/256/51/cc9bb4c82bc746/1a1a1aaa2198c627970773d80669d84574a8d80d3cb12453c02589f25382e353c32f94c33095fc5dc52a9c108ae956cf43ab/mhz_myth_320f.mov

(Wins the contest for longest URL at which this video is available)