There is one handout today at the front and back of the room!

Lecturer PSOE, new dad Dan Garcia

www.cs.berkeley.edu/~ddgarcia

Samsung pleads guilty! ⇒

They were convicted of “price-fixing” DRAM from 1999-04 to 2002-06 through emails, etc & ordered to pay $0.3 Billion (2\textsuperscript{nd} largest fine in criminal antitrust case).

www.cnn.com/2005/TECH/biztech/12/01/samsung.price.fixing.ap
Review

- Protocol suites allow heterogeneous networking
  - Another form of principle of abstraction
  - Protocols ↦ operation in presence of failures
  - Standardization key for LAN, WAN

- Magnetic Disks continue rapid advance: 60%/yr capacity, 40%/yr bandwidth, slow on seek, rotation improvements, MB/$ improving 100%/yr?
  - Designs to fit high volume form factor
State of the Art: Two camps (2005)

• Performance
  • Enterprise apps, servers
  • E.g., Seagate Cheetah 15K.4
    • Serial-Attached SCSI, Ultra320 SCSI, 2Gbit Fibre Channel interface
    • 146 GB, 3.5-inch disk
    • 15,000 RPM
    • 4 discs, 8 heads
    • 13 watts (idle)
    • 3.5 ms avg. seek
    • 200 MB/s transfer rate
    • 1.4 Million hrs MTBF
    • 5 year warrantee
    • $1000 = $6.8 / GB

• Capacity
  • Mainstream, home uses
  • E.g., Seagate Barracuda 7200.9
    • Serial ATA 3Gb/s, Ultra ATA/100
    • 500 GB, 3.5-inch disk
    • 7,200 RPM
    • ? discs, ? heads
    • 7 watts (idle)
    • 8.5 ms avg. seek
    • 300 MB/s transfer rate
    • ? Million hrs MTBF
    • 5 year warrantee
    • $330 = $0.66 / GB

source: www.seagate.com
1 inch disk drive!

- **2005 Hitachi Microdrive:**
  - 40 x 30 x 5 mm, 13g
  - 8 GB, 3600 RPM, 1 disk, 10 MB/s, 12 ms seek
  - 400G operational shock, 2000G non-operational
  - Can detect a fall in 4” and retract heads to safety
  - For iPods, cameras, phones

- **2006 MicroDrive?**
  - 16 GB, 12 MB/s!
  - Assuming past trends continue

www.hitachigst.com
Where does Flash memory come in?

• Microdrives and Flash memory (e.g., CompactFlash) are going head-to-head
  • Both non-volatile (no power, data ok)
  • Flash benefits: durable & lower power (no moving parts)
  • Flash limitations: finite number of write cycles (wear on the insulating oxide layer around the charge storage mechanism)
    - OEMs work around by spreading writes out

• How does Flash memory work?
  • NMOS transistor with an additional conductor between gate and source/drain which “traps” electrons. The presence/absence is a 1 or 0.
  • wikipedia.org/wiki/Flash_memory
### What does Apple put in its iPods?

<table>
<thead>
<tr>
<th>Type</th>
<th>Toshiba 0.5,1GB flash</th>
<th>Samsung 2,4GB flash</th>
<th>Hitachi 1 inch 4,6GB MicroDrive</th>
<th>Toshiba 1.8-inch 30,60GB (MK1504GAL)</th>
</tr>
</thead>
</table>

**Thanks to Andy Dahl for the tip**
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

Low End → High End
Replace Small Number of Large Disks with Large Number of Small Disks! (1988 Disks)

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

Disk Arrays potentially high performance, high MB per cu. ft., high MB per KW, \textbf{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 too unreliable to be useful!
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
  • Motivation: In the 1980s, there were 2 classes of drives: expensive, big for enterprises and small for PCs. They thought “make one big out of many small!”
  • Higher performance with more disk arms per $
  • Adds option for small # of extra disks (the “R”)
  • Started @ Cal by CS Profs Katz & Patterson
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 > $32 billion dollar industry, 80% nonPC disks sold in RAIDs
“RAID 0”: No redundancy = “AID”

• 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 data

- 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 (RAID 2 has bit-level striping)

• 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
RAID products: Software, Chips, Systems

RAID was $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…
Peer Instruction

1. RAID 1 (mirror) and 5 (rotated parity) help with performance and availability.

2. RAID 1 has higher cost than RAID 5.

3. Small writes on RAID 5 are slower than on RAID 1.

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

1. All RAID (0-5) helps with performance, only RAID 0 doesn’t help availability. TRUE

2. Surely! Must buy 2x disks rather than 1.25x (from diagram, in practice even even less) TRUE

3. RAID5 (2R,2W) vs. RAID1 (2W). Latency worse, throughput (|| writes) better. TRUE

1. RAID 1 (mirror) and 5 (rotated parity) help with performance and availability

2. RAID 1 has higher cost than RAID 5

3. Small writes on RAID 5 are slower than on RAID 1
**Administrivia**

- Please attend Wednesday’s lecture!
  - HKN Evaluations at the end

- Compete in the Performance contest!
  - Deadline is Mon, 2005-12-12 @ 11:59pm, 1 week from now

- Sp04 Final exam + solutions online!

- Final Review: 2005-12-11 @ 2pm in 10 Evans

- Final: 2005-12-17 @ 12:30pm in 2050 VLSB

- Only bring pen{,cil}s, two 8.5”x11” handwritten sheets + green. Leave backpacks, books.
## Upcoming Calendar

<table>
<thead>
<tr>
<th>Week #</th>
<th>Mon</th>
<th>Wed</th>
<th>Thu Lab</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td>#15 Last Week o’ Classes</td>
<td>Performance</td>
<td>LAST CLASS Summary, Review, &amp; HKN Evals</td>
<td>I/O Networking &amp; 61C Feedback Survey</td>
<td></td>
</tr>
<tr>
<td>#16 Sun 2pm Review 10 Evans</td>
<td>Performance competition due tonight @ midnight</td>
<td></td>
<td></td>
<td>FINAL EXAM SAT 12-17 @ 12:30pm-3:30pm 2050 VLSB Performance awards</td>
</tr>
</tbody>
</table>
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”

<table>
<thead>
<tr>
<th>Plane</th>
<th>DC to Paris</th>
<th>Top Speed</th>
<th>Passengers</th>
<th>Throughput (pmph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boeing 747</td>
<td>6.5 hours</td>
<td>610 mph</td>
<td>470</td>
<td>286,700</td>
</tr>
<tr>
<td>BAD/Sud Concorde</td>
<td>3 hours</td>
<td>1350 mph</td>
<td>132</td>
<td>178,200</td>
</tr>
</tbody>
</table>

- 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
  • performance(x) = \frac{1}{\text{execution\_time}(x)}

"F(ast) is \(n\) times faster than S(low)" means...

\[
\frac{\text{performance}(F)}{\text{execution\_time}(S)} = \frac{\text{performance}(S)}{\text{execution\_time}(F)}
\]
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 \( \Rightarrow \) 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 a program**

  \[ \text{CPU execution time} = \text{Clock Cycles for a program} \times \text{Clock Cycle Time} \]

• or

  \[ \text{CPU execution time} = \frac{\text{Clock Cycles for a program}}{\text{Clock Rate}} \]
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)

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

- 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: Execution Time / Clock cycle time
    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)

<table>
<thead>
<tr>
<th>Op</th>
<th>Freq(_i)</th>
<th>CPI(_i)</th>
<th>Prod</th>
<th>(% Time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>50%</td>
<td>1</td>
<td>.5</td>
<td>(23%)</td>
</tr>
<tr>
<td>Load</td>
<td>20%</td>
<td>5</td>
<td>1.0</td>
<td>(45%)</td>
</tr>
<tr>
<td>Store</td>
<td>10%</td>
<td>3</td>
<td>.3</td>
<td>(14%)</td>
</tr>
<tr>
<td>Branch</td>
<td>20%</td>
<td>2</td>
<td>.4</td>
<td>(18%)</td>
</tr>
</tbody>
</table>

**Instruction Mix**

\[
\text{Instruction Mix} = \frac{2.2}{2.2} = 1
\]

(Where time spent)

- What if Branch instructions twice as fast?
“And in conclusion…”

• RAID
  • Motivation: In the 1980s, there were 2 classes of drives: expensive, big for enterprises and small for PCs. They thought “make one big out of many small!”
  • Higher performance with more disk arms/$, adds option for small # of extra disks (the R)
  • Started @ Cal by CS Profs Katz & Patterson

• 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 processor design (datapath, pipelining effectiveness, caches, etc.)

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