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

Corrections to Lecture 42, pt I

• I said “Reel to reel is 15 & 30 feet/sec”. I meant to say “15 & 30 inches/Sec”
• Drives inside the iPod and iPod Mini:

Thanks to Andy Dahl for the tip

Hitachi 1 inch 4GB MicroDrive
Toshiba 1.8-inch 15/20/40GB (MK1504GAL)

Corrections to Lecture 42, pt II

• I said “AV drives put high-throughput data in the outer tracks”. From an APS Catalog:
  “Capturing AV data on a hard disk drive requires different performance characteristics than are required of a hard disk drive used for regular, everyday computer use. AV data capture involves long strings of data being input to and output from the drive. Severe data loss could occur if the drive is not prepared for it. High-capacity drives must go through some sort of thermal calibration cycle so that as the drive operates, and operating temperatures elevate, minute changes in the relationship between the location of the data heads and the data platter surfaces is updated. This calibration insures that the drive can find data in the location where it is expected, insuring optimal performance and data integrity. Non-AV drives often enter a calibration cycle on a regular schedule regardless of what the computer and the drive happen to be doing. In the case of an AV application, a drive entering a thermal calibration cycle (T-cal) will miss recording data. This might be represented by video capture that drops a few frames in the middle of an action sequence, or an audio track that seems to skip a beat. AV drives handle T-cal by rescheduling or postponing it until such time that the drive is not actively capturing data.”

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...


**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
  - \[ \text{performance}(x) = \frac{1}{\text{execution time}(x)} \]

  - “F(ast) is \( n \) times faster than S(low)” means...
    \[ \frac{\text{performance}(F)}{\text{execution time}(S)} = n \frac{\text{performance}(S)}{\text{execution time}(F)} \]

**Example of Response Time v. Throughput**

- Time of Concorde vs. Boeing 747?
  - Concorde 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
- Concorde 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 will 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 program**
  \[ \text{CPU time} = \text{Clock Cycles for a program} \times \text{Clock Cycle Time} \]
- or
  \[ \text{CPU time} = \text{Clock Cycles for a program} = \frac{\text{Clock Cycles for a program} \times \text{Clock Rate}}{\text{CPU execution time for program}} \]

Measuring Time using Clock Cycles (2/2)

- One way to define clock cycles:
  - **Clock Cycles for program**
    \[ = \text{Instructions for a program} \times \text{Average Clock cycles Per Instruction} \]
  (called “**Instruction Count**”)
  - 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**
  \[ \text{CPU time} = \frac{\text{Instruction Count}}{\text{CPI}} \times \text{Clock Cycle Time} \]
- Substituting for clock cycles:
  \[ \text{CPU time} = \frac{\text{Instruction Count}}{\text{CPI}} \times \text{Clock Cycle Time} \]

Performance Calculation (2/2)

- **CPU time** = \( \text{Instructions} \times \frac{\text{Cycles}}{\text{Program}} \times \frac{\text{Seconds}}{\text{Instruction}} \times \frac{\text{Cycle}}{\text{Instruction}} \times \frac{\text{Second}}{\text{Program}} \)
- 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: \( \frac{\text{Execution Time}}{\text{Clock cycle time}} \times \text{Instruction Count} \)
  - 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)

<table>
<thead>
<tr>
<th>Op</th>
<th>Freq</th>
<th>CPI</th>
<th>Prod (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>50%</td>
<td>1.5</td>
<td>(23%)</td>
</tr>
<tr>
<td>Load</td>
<td>20%</td>
<td>5.0</td>
<td>(45%)</td>
</tr>
<tr>
<td>Store</td>
<td>10%</td>
<td>3.0</td>
<td>(14%)</td>
</tr>
<tr>
<td>Branch</td>
<td>20%</td>
<td>2.0</td>
<td>(18%)</td>
</tr>
</tbody>
</table>

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 it’s hard to do
  - Don’t have access to machine to “benchmark” before purchase
  - Don’t know workload in future
- Friday: 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

“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.)

CPU time = Instructions x Cycles x Seconds