IC220 Set #18: Storage and I/O (Chapter 6)

ADMIN

- Reading – Chapter 6
  - Including RAID (6.9) but don’t stress memorizing the names of the levels
  - Can skip 6.10 and 6.11

I/O

- Important but neglected
  - “The difficulties in assessing and designing I/O systems have often relegated I/O to second class status”
  - “courses in every aspect of computing, from programming to computer architecture often ignore I/O or give it scanty coverage”
  - “textbooks leave the subject to near the end, making it easier for students and instructors to skip it!”
- GUILTY!
  - we won’t be looking at I/O in much detail
  - be sure and read Chapter 6 carefully
  - Later – IC322: Computer Networks
Outline

A. Overview
B. Physically connecting I/O devices to Processors and Memory (8.4)
C. Interfacing I/O devices to Processors and Memory (8.5)
D. Performance Measures (8.6)
E. Disk details/RAID (8.2)

(A) I/O Overview

• Can characterize devices based on:
  1. behavior
  2. partner (who is at the other end?)
  3. data rate
• Performance factors:
  — access latency
  — throughput
  — connection between devices and the system
  — the memory hierarchy
  — the operating system
• Other issues:
  — Expandability, dependability

(B) Connecting the Processor, Memory, and other Devices

Two general strategies:
1. Bus: _______ communication link
   Advantages:
   Disadvantages:
2. Point to Point Network: _______ links
   Use switches to enable multiple connections
   Advantages:
   Disadvantages:

Typical x86 PC I/O System
(B) Bus Basics – Part 1

- Types of buses:
  - Processor-memory
    - Short, high speed, fixed device types
    - custom design
  - I/O
    - lengthy, different devices
    - Standards-based e.g., USB, Firewire
    - Connect to proc-memory bus rather than directly to processor
- Only one pair of devices (sender & receiver) may use bus at a time
  - Bus ____________ decides who gets the bus next based on
    some ____________ strategy
  - May incorporate priority, round-robin aspects
- Have two types of signals:
  - “Data” – data or address
  - Control

I/O Bus Examples

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Firewire</th>
<th>USB 2.0</th>
<th>PCI Express</th>
<th>Serial ATA</th>
<th>Serial Attached SCSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intended use</td>
<td>External</td>
<td>External</td>
<td>Internal</td>
<td>Internal</td>
<td>External</td>
</tr>
<tr>
<td>Devices per channel</td>
<td>63</td>
<td>127</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Data width</td>
<td>4</td>
<td>2</td>
<td>2/lane</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Peak bandwidth</td>
<td>50MB/s</td>
<td>0.2MB/s</td>
<td>0.5MB/s, or 60MB/s</td>
<td>250MB/s, 5x, 2x, 4x, 1x, 16x, 32x</td>
<td>300MB/s, 300MB/s</td>
</tr>
<tr>
<td>Hot pluggable</td>
<td>Yes</td>
<td>Yes</td>
<td>Depends</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Max length</td>
<td>4.5m</td>
<td>5m</td>
<td>0.5m</td>
<td>1m</td>
<td>8m</td>
</tr>
<tr>
<td>Standard</td>
<td>IEEE 1394</td>
<td>USB Implementers Forum</td>
<td>PCI-SIG</td>
<td>SATA-IO</td>
<td>INCITS TC T10</td>
</tr>
</tbody>
</table>

(B) Bus Basics – Part 2

- Clocking scheme:
  1. Use a clock, signals change only on clock edge
     + Fast and small
     - All devices must operate at same rate
     - Requires bus to be short (due to clock skew)
  2. No clock, instead use “handshaking”
     + Longer buses possible
     + Accommodate wide range of device
     - More complex control

Handshaking example – CPU read from memory

1. CPU requests read
2. Memory acknowledges, CPU deasserts request
3. Memory sees change, deasserts Ack
4. Memory provides data, asserts DataRdy
5. CPU grabs data, asserts Ack
6. Memory sees Ack, deasserts DataRdy
7. CPU sees change, deasserts Ack
(C) Processor-to-device Communication

How does CPU send information to a device?
1. Special I/O instructions
   x86: inb / outb

   How to control access to I/O device?

2. Use normal load/instructions to special addresses
   Called ______________________
   Load/store put onto bus
   Memory ignores them (outside its range)
   Address may encode both device ID and a command

   How to control access to I/O device?

(C) Device-to-processor communication

How does device get data to the processor?
1. CPU periodically checks to see if device is ready: _______________
   • CPU sends request, keep checking if done
   • Or just checks for new info (mouse, network)

2. Device forces action by the processor when needed: _______________
   • Like an unscheduled procedure call
   • Same as “exception” mechanism that handles
     TLB misses, divide by zero, etc.

3. DMA:
   • Device sends data directly to memory w/o CPU’s involvement
   • Interrupts CPU when transfer is complete

(D) I/O’s impact on performance

• Total time = CPU time + I/O time

• Suppose our program is 90% CPU time, 10% I/O. If we improve CPU
  performance by 10x, but leave I/O unchanged, what will the new
  performance be?

• Old time = 100 seconds
• New time =

DMA Issues

What could go wrong?

- Processor
- Cache
- Main memory
- I/O controller
- Network
- Graphics output
- Disk

Memory - I/O bus

Interruption
(D) Measuring I/O Performance

- Latency?
- Throughput?
- Throughput with maximum latency?
- Transaction processing benchmarks
  - TPC-C
  - TPC-H
  - TPC-W
- File system / Web benchmarks
  - “Make” benchmark
  - SPECSFS
  - SPECWeb
  - SPECPower

(E) Disk Drives

- To access data:
  - seek: position head over the proper track (3 to 14 ms. avg.)
  - rotational latency: wait for desired sector (.5 / RPM)
  - transfer: grab the data (one or more sectors) 30 to 80 MB/sec

(E) RAID

- Idea: lots of cheap, smaller disks
- Small size and cost makes easier to add redundancy
- Multiple disks increases read/write bandwidth

RAID

RAID 0 – “striping”, no redundancy

RAID 1 – mirrored
RAID

**RAID 4 – Block-interleaved parity**

<table>
<thead>
<tr>
<th>Disk 0</th>
<th>Disk 1</th>
<th>Disk 2</th>
<th>Disk 3</th>
<th>Parity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block 1</td>
<td>Block 2</td>
<td>Block 3</td>
<td>Block 4</td>
<td>Parity 1</td>
</tr>
<tr>
<td>Block 5</td>
<td>Block 6</td>
<td>Block 7</td>
<td>Block 8</td>
<td>Parity 2</td>
</tr>
<tr>
<td>Block 9</td>
<td>Block 10</td>
<td>Block 11</td>
<td>Block 12</td>
<td>Parity 3</td>
</tr>
<tr>
<td>Block 13</td>
<td>Block 14</td>
<td>Block 15</td>
<td>Block 16</td>
<td>Parity 4</td>
</tr>
<tr>
<td>Parity 15-16</td>
<td>Block 17</td>
<td>Block 18</td>
<td>Block 19</td>
<td>Block 20</td>
</tr>
</tbody>
</table>

**RAID 5 – Distributed Block-interleaved Parity**

<table>
<thead>
<tr>
<th>Disk 0</th>
<th>Disk 1</th>
<th>Disk 2</th>
<th>Disk 3</th>
<th>Disk 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block 1</td>
<td>Block 2</td>
<td>Block 3</td>
<td>Block 4</td>
<td>Parity 5-4</td>
</tr>
<tr>
<td>Block 5</td>
<td>Block 6</td>
<td>Block 7</td>
<td>Block 8</td>
<td>Parity 5-4</td>
</tr>
<tr>
<td>Block 9</td>
<td>Block 10</td>
<td>Block 11</td>
<td>Block 12</td>
<td>Parity 5-4</td>
</tr>
<tr>
<td>Block 13</td>
<td>Block 14</td>
<td>Block 15</td>
<td>Block 16</td>
<td>Parity 5-4</td>
</tr>
<tr>
<td>Parity 17-18</td>
<td>Block 19</td>
<td>Block 20</td>
<td>Block 21</td>
<td>Block 22</td>
</tr>
</tbody>
</table>

Flash Storage

- Nonvolatile semiconductor storage
  - 100× – 1000× faster than disk
  - Smaller, lower power, more robust
  - But more $/GB (between disk and DRAM)

- Flash bits wears out after 1000’s of writes
  - Not suitable for direct RAM or disk replacement
  - Wear leveling: remap data to less used blocks
  - Result: “solid-state hard drive”

Fallacies and Pitfalls

- Fallacy: the rated mean time to failure of disks is 1,200,000 hours, so disks practically never fail.

- Fallacy: magnetic disk storage is on its last legs, will be replaced.

- Fallacy: A GB/sec bus can transfer 1 GB of data in 1 second.

- Pitfall: Moving functions from the CPU to the I/O processor, expecting to improve performance without analysis.