Lecture Outline

- HW5 and Project Questions?
- Storage and I/O
  - I/O Busses (7.3)
  - RAID (H&P 7.4-7.5)
I/O System Architecture

- **Buses**
  - Memory bus
  - I/O Bus
- **I/O processing**
  - Program controlled
  - DMA
  - I/O processors (IOPs)

Bus Issues

- **Clocking**: is bus clocked?
  - Synchronous: clocked, short bus or slow clock => fast
  - Asynchronous: no clock, use “handshaking” instead => slow
  - Isochronous: high-bandwidth, packet-based system (uniform in time)
- **Switching**: When control of bus is acquired and released
  - Atomic: bus held until request complete => slow
  - Split-transaction: bus free between request and reply => fast
- **Arbitration**: deciding who gets the bus next
  - Overlap arbitration for next master with current transfer
  - Daisy Chain: closer devices have priority => slow
  - Distributed: wired-OR, low-priority back-off => medium
- **Other issues**
  - Split data/address lines, width, burst transfer
Synchronous Data Transfer: Read Operation

Send the Address

Send Read Signal

Device Starts Sending Data

Device Says Data is Ready

CPU Reads Data

Asynchronous Data Transfer: Write Operation

t0: Master asserts lines

t1: Master waits and asserts Req

t2: Device asserts Ack (data recvd)

t3: Master releases Req (Handshake)

t4: Device releases Ack

Computer Science 146
David Brooks
When to use?

- When to use asynchronous vs. synchronous bus?
  - Mixed I/O speeds?
  - Bus length?
- Split transaction vs. atomic transaction?

---

### I/O and Memory Buses

<table>
<thead>
<tr>
<th></th>
<th>Bits</th>
<th>MHz</th>
<th>Peak MB/s</th>
<th>Special Features</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Memory Buses</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Summit</td>
<td>128</td>
<td>60</td>
<td>960</td>
<td></td>
</tr>
<tr>
<td>Challenge</td>
<td>256</td>
<td>48</td>
<td>1200</td>
<td></td>
</tr>
<tr>
<td>XDBus</td>
<td>144</td>
<td>66</td>
<td>1056</td>
<td></td>
</tr>
<tr>
<td><strong>I/O Buses</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISA</td>
<td>16</td>
<td>8</td>
<td>16</td>
<td>Original PC Bus</td>
</tr>
<tr>
<td>IDE/ATA</td>
<td>16</td>
<td>8-100</td>
<td>16-200</td>
<td>Disk, Tape, CD-ROM</td>
</tr>
<tr>
<td>PCI</td>
<td>32 (64)</td>
<td>33 (66)</td>
<td>133-333</td>
<td>“Plug + Play”</td>
</tr>
<tr>
<td>SCSI</td>
<td>8/16</td>
<td>5-160</td>
<td>10-320</td>
<td>High-level interface</td>
</tr>
<tr>
<td>PCMCIA</td>
<td>8/16</td>
<td>8</td>
<td>16</td>
<td>Modems, “hot-swap”</td>
</tr>
<tr>
<td>USB</td>
<td>Serial</td>
<td>A/Isoch.</td>
<td>1.5/60</td>
<td>Power line, packetized</td>
</tr>
<tr>
<td>FireWire</td>
<td>Serial</td>
<td>A/Isoch.</td>
<td>50/100</td>
<td>Fast USB</td>
</tr>
</tbody>
</table>

- Memory buses: speed (usually custom)
- I/O buses: compatibility (usually industry standard) + cost
Who Does I/O?

- **Main CPU**
  - Explicitly executes all I/O operations
    - Memory Mapped I/O
    - Special ISA I/O Operations (x86, IBM 370)
  - Interrupt Driven, Polling Based, or Hybrid (realtime)
    - High overhead, potential cache pollution
    + But no coherence problems
Assist the Main CPU

- **I/O Processor (IOP or channel processor)**
  - (special or general) processor dedicated to I/O operations
  + Fast
    - May be overkill, cache coherency problems
      - I/O sees stale data on output (memory not up to date)
      - CPU sees stale data in cache on input (I/O system only updates memory)
- **DMAC (direct memory access controller)**
  - Can transfer data to/from memory given start address (but that’s all)
  + Fast, usually simple
    - Still may be coherence problems, must be on memory bus

Communicating with I/O Processors

- Not issue if main CPU performs I/O by itself
- **I/O Control**: how to initialize DMAC/IOP?
  - Memory mapped: ld/st to preset, VM-protected address
  - Privileged I/O instructions
- **I/O Completion**: how does CPU know DMAC/IOP finished?
  - Polling: periodically check status bit => slow
  - Interrupt: I/O completion interrupts CPU => fast
- **Q**: *do DMAC/IOP use physical or virtual addresses?*
  - Physical: simpler, but can only transfer 1 page at a time
  - Virtual: More powerful, but DMAC/IOP needs address translation info
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

Advantages of Small Form Factor Disk Drives

Cost and Environmental Efficiencies
Replace Small Number of Large Disks with Large Number of Small 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.</td>
</tr>
<tr>
<td>Power</td>
<td>3 KW</td>
<td>11 W</td>
<td>1 KW</td>
</tr>
<tr>
<td>Data Rate</td>
<td>15 MB/s</td>
<td>1.5 MB/s</td>
<td>120 MB/s</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</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>

1988 Disk Drives vs. Disk Array

Disk Arrays have potential for large data and I/O rates, high MB per cu. ft., high MB per KW, but what about reliability?

Array Reliability

- Reliability of N disks = Reliability of 1 Disk ÷ N

50,000 Hours ÷ 70 disks = 700 hours

Disk system MTTF: Drops from 6 years to 1 month!

- Arrays (without redundancy) too unreliable to be useful!

Hot spares support reconstruction in parallel with access: very high media availability can be achieved

Computer Science 146
David Brooks
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

Redundant Arrays of Inexpensive Disks
RAID 1: Disk Mirroring/Shadowing

- Each disk is fully duplicated onto its “mirror”
  - Very high availability can be achieved
- Bandwidth sacrifice on write:
  - Logical write = two physical writes
- Reads may be optimized
- Most expensive solution: 100% capacity overhead
- Seek times can be optimized (choose disk with shortest seek)
  - (RAID 2 not interesting, so skip)
Redundant Array of Inexpensive Disks RAID 3: Parity Disk

- Sum computed across recovery group to protect against hard disk failures, stored in P disk
- Logically, a single high capacity, high transfer rate disk: good for large transfers
- Wider arrays reduce capacity costs, but decreases availability
- 33% capacity cost for parity in this configuration
Inspiration for RAID 4

- RAID 3 relies on parity disk to discover errors on Read
  - Every access goes to all disks
  - Some apps want to do smaller accesses, allowing independent accesses to occur in parallel
  - Independent small reads are ok because disk can detect errors
    - Every sector has an error detection field
  - Independent small writes are trickier – don’t we have to update parity?

Problems of Disk Arrays:
Small Writes on RAID3

RAID-3: Small Write Algorithm
1 Logical Write = 3 Physical Reads + 2 Physical Writes
Problems of Disk Arrays:
Small Writes on RAID4/5

RAID-5: Small Write Algorithm
1 Logical Write = 2 Physical Reads + 2 Physical Writes

RAID 4: High I/O Rate Parity

Increasing Logical Disk Address

Example: small read D0 & D5, large write D12-D15
Inspiration for RAID 5

- RAID 4 works well for small reads
- Small writes (write to one disk):
  - RAID 3: read other data disks, create new sum and write to Parity Disk
  - RAID 4/5: since P has old sum, compare old data to new data, add the difference to P
- Small writes are limited by Parity Disk: Write to D0, D5 both also write to P disk (Parity Disk Bottleneck)

RAID 5: High I/O Rate Interleaved Parity

Independent writes possible because of interleaved parity

Example: write to D0, D5 uses disks 0, 1, 3, 4
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 a $19 billion dollar industry, 80% of non-PC disks sold in RAIDs.

### RAID in Industry

<table>
<thead>
<tr>
<th>RAID Level</th>
<th>Name</th>
<th>Minimum Number of Disk Faults Survived</th>
<th>Example Data Disks</th>
<th>Corresponding Check Disks</th>
<th>Industry Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Nonredundant Striped</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>Widely Used</td>
</tr>
<tr>
<td>1</td>
<td>Mirrored</td>
<td>1</td>
<td>8</td>
<td>8</td>
<td>EMC, IBM Compaq</td>
</tr>
<tr>
<td>2</td>
<td>Memory-style ECC</td>
<td>1</td>
<td>8</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Bit-interleaved Parity</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>Storage Concepts</td>
</tr>
<tr>
<td>4</td>
<td>Block-interleaved Parity</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>Network Appliance</td>
</tr>
<tr>
<td>5</td>
<td>Block-interleaved distributed parity</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>Widely Used</td>
</tr>
<tr>
<td>6</td>
<td>P+Q redundancy</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Computer Science 146
David Brooks
Summary: RAID Techniques

Goal: Performance, popularity due to reliability of storage

- **Disk Mirroring, Shadowing (RAID 1)**
  - Each disk is fully duplicated onto its "shadow"
  - Logical write = two physical writes
  - 100% capacity overhead

- **Parity Data Bandwidth Array (RAID 3)**
  - Parity computed horizontally
  - Logically a single high data bw disk

- **High I/O Rate Parity Array (RAID 5)**
  - Interleaved parity blocks
  - Independent reads and writes
  - Logical write = 2 reads + 2 writes

---

I/O System Example

- Given
  - 500 MIPS CPU
  - 16B wide, 100 ns memory system
  - 10000 instrs per I/O
  - 16KB per I/O
  - 200 MB/s I/O bus, with room for 20 SCSI-2 controllers
  - SCSI-2 strings (buses) – 20MB/s with 15 disks per bus
  - SCSI-2 1ms overhead per I/O
  - 7200 RPM (120 RPS), 8ms avg seek, 6MB/s transfer disks
  - 200 GB total storage

- Q: Choose 2GB or 8GB disks for maximum IOPS?
  - How to arrange disks and controllers?
I/O System Example (cont’d)

• Step 1: Calculate CPU, memory, I/O bus peak IOPS
  – CPU: $500 \text{ MIPS/10000 instructions/IO} = 50000 \text{ IOPS}$
  – Memory: $(16\text{-bytes / 100ns}) / 16\text{KB} = 10000 \text{ IOPS}$
  – I/O bus: $(200 \text{MB/s}) 16\text{KB} = 12500 \text{ IOPS}$
  – Memory bus (10000 IOPS) is the bottleneck

• Step 2: Calculate Disk IOPS
  – $T_{\text{disk}} = 8\text{ms} + 0.5/120 \text{ RPS} + 16\text{KB/(6MB/s)} = 15 \text{ ms}$
  – Disk: $1 / 15\text{ms} = 67 \text{ IOPS}$
  – 8GB Disks => need 25 => $25 \times 67 \text{ IOPS} = 1675 \text{ IOPS}$
  – 2GB Disks => need 100 => $100 \times 67 \text{ IOPS} = 6700 \text{ IOPS}$
  – 100 2GB disks (6700 IOPS) are new bottleneck

• Answer: 100 2GB disks!

I/O System Example (cont’d)

• Step 3: Calculate SCSI-2 controller peak IOPS
  – $T_{\text{SCSI-2}} = 1\text{ms} + 16\text{KB/(20MB/s)} = 1.8\text{ms}$
  – SCSI-2: $1/1.8\text{ms} = 556 \text{ IOPS}$

• Step 4: How many disks per controller?
  – $556 \text{ IOPS} / 67 \text{ IOPS} = 8 \text{ disks per controller}$

• Step 5: How many controllers?
  – 100 disks/ 8 disks/controller = 13 controllers

• Answer: 13 controllers, 8-disks each
Next Lecture

• Wednesday:
  – Google Cluster
  Reading:
  – Course Summary and Wrapup
  – Schedule a time for the Final Review