I/O Devices (CSE 422S)

Ken Wong
Washington University

kenw@wustl.edu
www.arl.wustl.edu/~kenw

I/O Architectures

I/O Operations (1)

- Programmed I/O (Direct I/O with Polling)
  - Direct I/O: Processor issues I/O command on behalf of a process
  - Poll: Process busy-waits for I/O completion
- Direct Memory Access (DMA)
  - A Specialized processor that transfers data between memory and I/O device while CPU does other tasks
  - Operation
    - OS loads DMA registers: Request type, Count, Buffer Address
    - Processor issues block I/O command to DMA module on behalf of a process
    - DMA module controls movement of data between main memory and controller
    - Processor continues with other instructions and is interrupted by controller when I/O completes

I/O Port (Unified approach to I/O programming)

- Set of addresses (perhaps mapped to memory)
- Control Register: Commands
- Status Register: Internal state of device
- Input Register: Data pulled from device
- Output Register: Data to be pushed to device
**I/O Operations (2)**

- **Interrupt-Driven I/O**
  - Processor issues I/O command on behalf of a process
  - Processor continues with other instructions and is interrupted by controller when I/O completes
  - Relieves CPU from waiting for every I/O event
    - Many CPU cycles still spent transferring data

**I/O Buffering Schemes**

- **Purpose:** Smooth out I/O traffic

- **No Buffering**
- **Single Buffering**
- **Double Buffering**
- **Ring Buffering**

**Simple Disk Geometry**

- Areal density (and capacity) has been doubling every 2-3 years

- **Areal Density** = \( BPI \times TPI \)
  - 2.5” or 3.5” platters
  - 2400, 5400, 5600, 6400, 7200, 10000 RPM

**Physical Disk Management**

- **Physical Formatting**
  - Create (on the disk surface) the electronic patterns that define the smallest data transfer unit (sector)
    - Header: Sector number, bad sector information
    - Body: Data (sector contents)
    - Trailer: Error detection/correction
  - Create an initial disk label
    - Define partitions (cylinder groupings)
    - Record disk geometry (# cylinders, # heads, # sectors/track)

- **Bad sector** (a sector with at least 1 bad bit)
  - Sector Forwarding: Chain bad sector to a sector in extra cylinder
  - Sector Slipping: Forward shift header information on a track
Disk Request Time

- **Wait for Device**
  - Wait for Channel
  - Seek (Move Arm)
  - Rotational Latency
  - Data Transfer
  - Time

- **Average Seek Time** \( t_{\text{seek}} = k_0 + n_{\text{cylinders}} k_1 \)
- **Average Rotational Latency (Delay)**
  \( t_{\text{rotation}} = T_r/2 \)
- **Average Transfer Time**
  \( t_{\text{transfer}} = T_r L_{\text{block}}/C_{\text{track}} \)
- **Average Access Time**
  \( t_{\text{access}} = t_{\text{seek}} + T_r/2 + T_r L_{\text{block}}/C_{\text{track}} \)

Disk Request Time Examples

- **Disk Parameters**
  - Average Seek Time: 20 ms
  - Rotation Speed: 7200 RPM (\( T_r = 8.35 \text{ ms rotation time} \))
  - 512-Byte sectors
  - 64 sectors per track \( C_{\text{track}} = 64 \times 512 \text{ bytes} \)

- **Average time to read 128K bytes (Read 4 whole tracks)**
  - Average Time (ms): 4 (4.18 + 8.35) = 50.12 ms

- **Average time to read 128K bytes (Read 256 random sectors)**
  - Average Time (ms): 256 (20 + 4.18 + 0.13) = 6223.36 ms = 6.2 sec

- **Significant Effect:** Location of the sectors

Zoned Recording

- The greater the distance from the center ➔
  - The higher the recording density
  - The higher the data transfer rate

- Typically, 10 or more zones
  - Transfer rate highest at outer track
  - All EIDE and SCSI drives are now zoned (Hidden from PC BIOS)

Quantum Fireball 3.8 GB Drive

<table>
<thead>
<tr>
<th>Zone</th>
<th>#Tracks</th>
<th>Sectors/Track Transfer Rate (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (outer)</td>
<td>454</td>
<td>232</td>
</tr>
<tr>
<td>4</td>
<td>454</td>
<td>214</td>
</tr>
<tr>
<td>9</td>
<td>454</td>
<td>170</td>
</tr>
<tr>
<td>14</td>
<td>454</td>
<td>122</td>
</tr>
</tbody>
</table>

Source: Scott Mueller, Upgrading and Repairing PCs

- All IDE and SCSI drives use zoned bit recording
- 6,810 tracks/surface
- Each surface has 1,259,396 sectors (615 MB)
- Disk controller built into drive
  - Drive appears to have same #sectors/track to PC BIOS
Disk Scheduling Policies

- Example (Start Cylinder = 100; Largest Cylinder # = 199)
  - Cylinder Numbers in Request Queue: 55, 58, 39, 18, 90, 160, 150, 38
- **FIFO** (First-In-First-Out): 55, 58, 39, 18, 90, 160, 150, 38
- **SSTF** (Shortest-Seek-Time-First): 90, 58, 55, 39, 38, 18, 150, 160
- **SCAN** (Elevator Algorithm): (right first) 150, 160, 199, 90, 58, 39, 38, 18, 0
  - Bidirectional: Increasing, goto max, decreasing, goto 0, ...
- **C-SCAN** (Circular SCAN): (right first) 150, 160, 199, 0, 18, 38, 39, 55, 58, 90
  - Unidirectional: Increasing, goto max, goto 0, increasing, ...
- **LOOK** (right first) 150, 160, 90, 58, 55, 39, 38, 18
  - Don’t go to extreme cylinders unless necessary
- **C-LOOK** (right first) 150, 160, 18, 38, 39, 55, 58, 90

Server Configurations

- Laboratory (circa 1983) - $250K
  - VAX 750 CPU (0.7 MIPS), 4 MB RAM, 2-250 MB CDC disk, 75 ips 1600 bpi tape, 10 Mbps Ethernet coax
- Laboratory (circa 1992) - $60K
  - Sun SPARC CPU, 16 MB RAM, 2-500 MB SCSI-1 disk, Exabyte tape, 10 Mbps Ethernet thin
- Neuroscience Data Cache (circa 2000) - $500K
  - Dual CPU SUN SPARCcenter, 1 GB RAM, 1.8 TB RAID-5 (48 x 9 GB + 24 x 18 GB + 24 x 36 GB SCSI-2, quad power, dual controller), 10/100 Mbps Ethernet cat5, 600 Mbps ATM fiber, remote DLT robot backup
- Laboratory (circa 2005) - $25K
  - Dual 3 GHz Xeon CPU, 2 GB RAM, 1.5 TB RAID-5, GigE copper, remote tape robot backup

RAID

- Redundant Array of Inexpensive Disks
- Set of physical disk drives seen by OS as a single logical drive
- Data distributed across physical drives of an array
- Redundant disk space is used to store parity information which guarantees data recoverability in case of disk failure
- Motivation
  - Replace large, expensive disk drives with multiple, less-expensive, smaller-capacity drives
- Tradeoff
  - Increase availability at the cost of increased unreliability (probability of a single failure)
**RAID Levels**

- **RAID 0**: Striping
- **RAID 1**: Mirroring
- **RAID 2**: Hamming Code
- **RAID 3**: Bit-Interleaved Parity
- **RAID 4**: Block-Interleaved Parity
- **RAID 5**: Distributed RAID 4

**RAID 0 (Striping)**

- Small versus large strip size? Concurrent transfer versus concurrent I/O requests.

**RAID 0**

- A strip is N sectors
- Data are striped across the disk drives
  - Example: Strips 0, 1, 2, 3 = Stripe 0
- Performance: Function of request pattern and data layout
- High Data Transfer Capacity
  - Need high transfer capacity from drive to host memory
  - Need requests to be for large amounts of logically contiguous data
    - Overlap seek and transfer times of all disk drives
    - Compare to 1 seek followed by large transfer while other drives are idle
- High I/O Request Rate
  - Initiate concurrent, independent requests (each reading 1 strip)
- Disadvantage: One drive fails → Array is useless

**Effect of Striping**

- Two 1-stripe accesses (1 stripe = 1 block)
  - Non-RAID (1 Large Disk)
  - RAID 0 (2 Disks)
- One 2-stripe access (1 stripe = 1 block)
  - Non-RAID (1 Large Disk)
  - RAID 0 (2 Disks)
**RAID 1 (Mirroring)**

- **Pros**
  - Easy to implement
  - Read Request: Service from disk drive which minimizes service time
  - Write Request: Update strip and its mirror in parallel
  - Simple recovery from disk failure: Access disk mirror
  - Very good data reliability
  - Improves read performance

- **Cons**
  - 100% storage overhead ➔ High byte cost

**Case for RAID 1**

- Storage costs are dropping; Down time cost is rising.

**RAID 3 (Bit-Interleaved Parity)**

- Stripe bits or bytes over disks and compute parity over stripe
- Simple parity bit computed for same bit position on all drives
  - For m=4 drives, parity bit i on drive 4 (the parity drive):
    \[ X4(i) = X0(i) \oplus X1(i) \oplus X2(i) \oplus X3(i) \]
  - 1 parity drive, no matter how large the disk array
  - Parallel access with data distributed in small strips

**Recovery From Failure**

- Reconstruct data from remaining drives until failed drive is replaced
- Example: Drive X1 fails when m=4 drives
  \[
  X1(i) = X0(i) \oplus X2(i) \oplus X3(i) \oplus X4(i) \\
  X4(i) \oplus X4(i) \oplus X1(i) = X0(i) \oplus X1(i) \oplus X2(i) \oplus X3(i) \oplus X4(i) \oplus X1(i) 
  \]
### RAID 3

**Write Request**
- Example: \( m=4 \) drives and update strip \( y \) on drive \( X1 \)
  - Data drives: \( X0, X1, X2, X3 \)
  - Parity drive: \( X4 \)
- Operation
  - Summary: Parallel access of small strip \( y \) from each disk
  - Read strip \( y \) from all drives
  - Update strip \( y \) from drive \( X1 \)
  - Compute strip \( y \) for drive \( X4 \)
  - Write back strip \( y \) of all drives

**Con:**
- Every read or write needs to access all drives of a set ➔ Only 1 pending request per disk set

### RAID 4 AND 5 (Block-level Parity)

**RAID 4:** Parity Disk
- \((\text{Bit } i, \text{ Block } j) = f(\text{Bit } i, \text{ Block } j, \text{ all data disks})\)
- Stripe blocks over disks and compute block parity over stripe of blocks
- Can read a single block in a stripe (independent drives)
- Write requires waiting for parity block(s) to be written
  - Parity drive becomes a bottleneck

**RAID 5:** Distribute parity blocks among all disks (Avoid parity disk bottleneck)
- The most popular type today
- Very good read performance
- Write performance is better than RAID 4, but is still slow because of parity block.