Chapter 6: I/O & Storage

Instructor: Huzefa Rangwala, PhD
Big Picture: Where are We Now?

- The five classic components of a computer

```
Processor
  Control
  Datapath

Memory

Input
Output
```

- Topics:
  - Storage system
  - BUS
Introduction

I/O devices can be characterized by
- Behaviour: input, output, storage
- Partner: human or machine
- Data rate: bytes/sec, transfers/sec

I/O bus connections

![Diagram of I/O bus connections]

Chapter 6 — Storage and Other I/O Topics — 3
I/O System Characteristics

- Dependability is important
  - Particularly for storage devices

- Performance measures
  - Latency (response time)
  - Throughput (bandwidth)

- Desktops & embedded systems
  - Mainly interested in response time & diversity of devices

- Servers
  - Mainly interested in throughput & expandability of devices
Fault: failure of a component
  - May or may not lead to system failure
Dependability Measures

- Reliability: mean time to failure (MTTF)
- Service interruption: mean time to repair (MTTR)
- Mean time between failures
  - $MTBF = MTTF + MTTR$
- Availability = $\frac{MTTF}{MTTF + MTTR}$
- Improving Availability
  - Increase MTTF: fault avoidance, fault tolerance, fault forecasting
  - Reduce MTTR: improved tools and processes for diagnosis and repair
Magnetic Disks

- Magnetic disks still play the central role in storage systems
  - Inexpensive
  - Nonvolatile: DRAM alone cannot replace disk
  - Relatively fast: compared to tape, recordable CD
    - But much slower than DRAM
## Disk History

<table>
<thead>
<tr>
<th>Year</th>
<th>Data Density (Mbit/sq. in.)</th>
<th>Capacity (MBytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1973</td>
<td>1.7</td>
<td>140</td>
</tr>
<tr>
<td>1979</td>
<td>7.7</td>
<td>2,300</td>
</tr>
</tbody>
</table>

Disk History

1989:
63 Mbit/sq. in
60,000 MBytes

1997:
1450 Mbit/sq. in
2300 MBytes

1997:
3090 Mbit/sq. in
8100 MBytes

“Makers of disk drives crowd even more data into even smaller spaces”
Disk Storage

- Nonvolatile, rotating magnetic storage
Disk Sectors and Access

- Each sector records
  - Sector ID
  - Data (512 bytes, 4096 bytes proposed)
  - Error correcting code (ECC)
    - Used to hide defects and recording errors
  - Synchronization fields and gaps

- Access to a sector involves
  - Queuing delay if other accesses are pending
  - Seek: move the heads
  - Rotational latency
  - Data transfer
  - Controller overhead
Disk Access Example

- **Given**
  - 512B sector, 15,000rpm, 4ms average seek time, 100MB/s transfer rate, 0.2ms controller overhead, idle disk

- **Average read time**
  - 4ms seek time
    + $\frac{1}{2} / (15,000/60) = 2\text{ms}$ rotational latency
    + $512 / 100\text{MB/s} = 0.005\text{ms}$ transfer time
    + 0.2ms controller delay
    = 6.2ms

- **If actual average seek time is 1ms**
  - Average read time = 3.2ms
Disk Performance Issues

• Manufacturers quote average read time
  ◦ Based on all possible seeks
  ◦ Locality and OS scheduling lead to smaller actual average seek times

• Smart disk controller allocate physical sectors on disk
  ◦ Present logical sector interface to host
  ◦ SCSI, ATA, SATA

• Disk drives include caches
  ◦ Pre-fetch sectors in anticipation of access
  ◦ Avoid seek and rotational delay
Flash Storage

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

• NOR flash: bit cell like a NOR gate
  ◦ Random read/write access
  ◦ Used for instruction memory in embedded systems

• NAND flash: bit cell like a NAND gate
  ◦ Denser (bits/area), but block-at-a-time access
  ◦ Cheaper per GB
  ◦ Used for USB keys, media storage, …

• Flash bits wears out after 1000’s of accesses
  ◦ Not suitable for direct RAM or disk replacement
  ◦ Wear leveling: remap data to less used blocks
Interconnecting Components

- Need interconnections between
  - CPU, memory, I/O controllers

- Bus: shared communication channel
  - Parallel set of wires for data and synchronization of data transfer
  - Can become a bottleneck

- Performance limited by physical factors
  - Wire length, number of connections

- More recent alternative: high-speed serial connections with switches
  - Like networks
Bus Types

- Processor-Memory buses
  - Short, high speed
  - Design is matched to memory organization

- I/O buses
  - Longer, allowing multiple connections
  - Specified by standards for interoperability
  - Connect to processor-memory bus through a bridge
Bus Signals and Synchronization

- **Data lines**
  - Carry address and data
  - Multiplexed or separate

- **Control lines**
  - Indicate data type, synchronize transactions

- **Synchronous**
  - Uses a bus clock

- **Asynchronous**
  - Uses request/acknowledge control lines for handshaking
## I/O Bus Examples

<table>
<thead>
<tr>
<th></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><strong>Intended use</strong></td>
<td>External</td>
<td>External</td>
<td>Internal</td>
<td>Internal</td>
<td>External</td>
</tr>
<tr>
<td><strong>Devices per channel</strong></td>
<td>63</td>
<td>127</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td><strong>Data width</strong></td>
<td>4</td>
<td>2</td>
<td>2/lane</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td><strong>Peak bandwidth</strong></td>
<td>50MB/s or 100MB/s</td>
<td>0.2MB/s, 1.5MB/s, or 60MB/s</td>
<td>250MB/s/lane 1×, 2×, 4×, 8×, 16×, 32×</td>
<td>300MB/s</td>
<td>300MB/s</td>
</tr>
<tr>
<td><strong>Hot pluggable</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Depends</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Max length</strong></td>
<td>4.5m</td>
<td>5m</td>
<td>0.5m</td>
<td>1m</td>
<td>8m</td>
</tr>
<tr>
<td><strong>Standard</strong></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>
I/O Management

• I/O is mediated by the OS
  ◦ Multiple programs share I/O resources
    • Need protection and scheduling
  ◦ I/O causes asynchronous interrupts
    • Same mechanism as exceptions (handled by OS)
  ◦ I/O programming is fiddly
    • OS provides abstractions to programs
I/O Commands

- I/O devices are managed by I/O controller hardware
  - Transfers data to/from device
  - Synchronizes operations with software
- Command registers
  - Cause device to do something
- Status registers
  - Indicate what the device is doing and occurrence of errors
- Data registers
  - Write: transfer data to a device
  - Read: transfer data from a device
Memory-Mapped I/O

An I/O scheme in which portions of address space are assigned to I/O devices, and reads and writes to those addresses are commands to the I/O device.
I/O Register Mapping

- Memory mapped I/O
  - Registers are addressed in same space as memory
  - Address decoder distinguishes between them
  - OS uses address translation mechanism to make them only accessible to kernel
Dedicated I/O instruction

- I/O instructions (2nd way of device communication)
  - Separate instructions to access I/O registers
  - Can only be executed in kernel mode
  - Example: x86
Polling

- Periodically check I/O status register
  - If device ready, do operation
  - If error, take action

- Common in small or low-performance real-time embedded systems
  - Predictable timing
  - Low hardware cost

- In other systems, wastes CPU time
Interrupts

- When a device is ready or error occurs
  - Controller interrupts CPU
- Interrupt is like an exception
  - But not synchronized to instruction execution
  - Can invoke handler between instructions
  - Cause information often identifies the interrupting device

- Priority interrupts
  - Devices needing more urgent attention get higher priority
  - Can interrupt handler for a lower priority interrupt
I/O Data Transfer

- **Polling and interrupt-driven I/O**
  - **CPU** transfers data between memory and I/O data registers
  - Time consuming for high-speed devices

- **Direct memory access (DMA)**
  - OS provides starting address in memory
  - I/O controller transfers to/from memory autonomously
  - Controller interrupts on completion or error but not for transfer
DMA

- Processor Sets up the DMA
  - Identity of the device
  - Operation on the device
  - Memory address of source – destination
  - Number of bytes to be transferred

- DMA operates
  - Starts device – waits for data (memory or device)

- DMA completes – controller interrupts the CPU, which can interrogate DMA or memory.
DMA/Cache Interaction

- If DMA writes to a memory block that is cached
  - Cached copy becomes stale
- If write-back cache has dirty block, and DMA reads memory block
  - Reads stale data
- Need to ensure cache coherence
  - Flush blocks from cache if they will be used for DMA
  - Or use non-cacheable memory locations for I/O
DMA/VM Interaction

- OS uses virtual addresses for memory
  - DMA blocks may not be contiguous in physical memory
- Should DMA use virtual addresses?
  - Would require controller to do translation
- If DMA uses physical addresses
  - May need to break transfers into page-sized chunks
  - Or chain multiple transfers
  - Or allocate contiguous physical pages for DMA
Measuring I/O Performance

- I/O performance depends on
  - Hardware: CPU, memory, controllers, buses
  - Software: operating system, database management system, application
  - Workload: request rates and patterns

- I/O system design can trade-off between response time and throughput
  - Measurements of throughput often done with constrained response-time
Transaction Processing Benchmarks

- **Transactions**
  - Small data accesses to a DBMS
  - Interested in I/O rate, not data rate

- **Measure throughput**
  - Subject to response time limits and failure handling
  - ACID (Atomicity, Consistency, Isolation, Durability)
  - Overall cost per transaction

- **Transaction Processing Council (TPC) benchmarks** ([www.tcp.org](http://www.tcp.org))
  - TPC-APP: B2B application server and web services
  - TCP-C: on-line order entry environment
  - TCP-E: on-line transaction processing for brokerage firm
  - TPC-H: decision support — business oriented ad-hoc queries
File System & Web Benchmarks

- **SPEC System File System (SFS)**
  - Synthetic workload for NFS server, based on monitoring real systems
  - Results
    - Throughput (operations/sec)
    - Response time (average ms/operation)

- **SPEC Web Server benchmark**
  - Measures simultaneous user sessions, subject to required throughput/session
  - Three workloads: Banking, Ecommerce, and Support
I/O vs. CPU Performance

Amdahl’s Law
- Don’t neglect I/O performance as parallelism increases compute performance

Example
- Benchmark takes 90s CPU time, 10s I/O time
- Double the number of CPUs/2 years
  - I/O unchanged

<table>
<thead>
<tr>
<th>Year</th>
<th>CPU time</th>
<th>I/O time</th>
<th>Elapsed time</th>
<th>% I/O time</th>
</tr>
</thead>
<tbody>
<tr>
<td>now</td>
<td>90s</td>
<td>10s</td>
<td>100s</td>
<td>10%</td>
</tr>
<tr>
<td>+2</td>
<td>45s</td>
<td>10s</td>
<td>55s</td>
<td>18%</td>
</tr>
<tr>
<td>+4</td>
<td>23s</td>
<td>10s</td>
<td>33s</td>
<td>31%</td>
</tr>
<tr>
<td>+6</td>
<td>11s</td>
<td>10s</td>
<td>21s</td>
<td>47%</td>
</tr>
</tbody>
</table>
RAID

- Redundant Array of Inexpensive (Independent) Disks
  - Use multiple smaller disks (c.f. one large disk)
  - Parallelism improves performance
  - Plus extra disk(s) for redundant data storage
- Provides fault tolerant storage system
  - Especially if failed disks can be “hot swapped”
- RAID 0
  - No redundancy (“AID”?)
    - Just stripe data over multiple disks
  - But it does improve performance
### RAID 1 & 2

- **RAID 1: Mirroring**
  - N + N disks, replicate data
    - Write data to both data disk and mirror disk
    - On disk failure, read from mirror

- **RAID 2: Error correcting code (ECC)**
  - N + E disks (e.g., 10 + 4)
  - Split data at bit level across N disks
  - Generate E-bit ECC
  - Too complex, not used in practice
RAID 3: Bit-Interleaved Parity

- \( N + 1 \) disks
  - Data striped across \( N \) disks at byte level
  - Redundant disk stores parity
  - Read access
    - Read all disks
  - Write access
    - Generate new parity and update all disks
  - On failure
    - Use parity to reconstruct missing data

- Not widely used
RAID 4: Block-Interleaved Parity

- **N + 1 disks**
  - Data striped across N disks at block level
  - Redundant disk stores parity for a group of blocks
- **Read access**
  - Read only the disk holding the required block
- **Write access**
  - Just read disk containing modified block, and parity disk
  - Calculate new parity, update data disk and parity disk
- **On failure**
  - Use parity to reconstruct missing data

- Not widely used
RAID 3 vs RAID 4

New Data
1. Read
2. Read
3. Read

D0' D0 D1 D2 D3 P

XOR

D0' D1 D2 D3 P'

4. Write
5. Write

New Data
1. Read
2. Read

D0' D0 D1 D2 D3 P

XOR

D0' D1 D2 D3 P'

3. Write
4. Write
RAID 5: Distributed Parity

- N + 1 disks
  - Like RAID 4, but parity blocks distributed across disks
    - Avoids parity disk being a bottleneck
- Widely used
RAID 6: P + Q Redundancy

• N + 2 disks
  ◦ Like RAID 5, but two lots of parity
  ◦ Greater fault tolerance through more redundancy

• Multiple RAID
  ◦ More advanced systems give similar fault tolerance with better performance
RAID Summary

- RAID can improve performance and availability
  - High availability requires hot swapping

- Assumes independent disk failures
  - Too bad if the building burns down!

- See “Hard Disk Performance, Quality and Reliability”
I/O System Design

- Satisfying latency requirements
  - For time-critical operations
  - If system is unloaded
    - Add up latency of components

- Maximizing throughput
  - Find “weakest link” (lowest-bandwidth component)
  - Configure to operate at its maximum bandwidth
  - Balance remaining components in the system

- If system is loaded, simple analysis is insufficient
  - Need to use queuing models or simulation
Server Computers

- Applications are increasingly run on servers
  - Web search, office apps, virtual worlds, …
- Requires large data center servers
  - Multiple processors, networks connections, massive storage
  - Space and power constraints
- Server equipment built for 19” racks
  - Multiples of 1.75” (1U) high
Rack-Mounted Servers

Sun Fire x4150 1U server

- 2 Redundant power Supplies
- 3 PCI Express Slots
- System Status LEDs
- Management NIC
- 2 USB Ports
- 4 Gigabit NICs
- Video
Chapter 6 — Storage and Other I/O Topics — 47

Sun Fire x4150 1U server

4 cores each

16 x 4GB = 64GB DRAM
I/O System Design Example

- Given a Sun Fire x4150 system with
  - Workload: 64KB disk reads
    - Each I/O op requires 200,000 user-code instructions and 100,000 OS instructions
  - Each CPU: $10^9$ instructions/sec
  - FSB: 10.6 GB/sec peak (sustain half peak rate) [2 FSB]
  - DRAM DDR2 667MHz: 5.336 GB/sec
  - PCI-E 8× bus: $8 \times 250\text{MB/sec} = 2\text{GB/sec}$
  - Disks: 15,000 rpm, 2.9ms avg. seek time, 112MB/sec transfer rate, rotational latency = 4ms
  - For sequential access: just look at transfer rate.

- What I/O rate can be sustained?
  - For random reads, and for sequential reads
Design Example (cont)

- **I/O rate for CPUs**
  - Per core: \( \frac{10^9}{(100,000 + 200,000)} = 3,333 \)
  - 8 cores: 26,667 ops/sec

- **Random reads, I/O rate for disks**
  - Assume actual seek time is average/4
  - Time/op = seek + latency + transfer
    \[ = \frac{2.9\text{ms}}{4} + \frac{4\text{ms}}{2} + \frac{64\text{KB}}{112\text{MB/s}} = 3.3\text{ms} \]
  - 303 ops/sec per disk, 2424 ops/sec for 8 disks

- **Sequential reads**
  - \( \frac{112\text{MB/s}}{64\text{KB}} = 1750 \text{ ops/sec per disk} \)
  - 14,000 ops/sec for 8 disks
Design Example (cont)

- PCI-E I/O rate
  - $2\text{GB/sec} / 64\text{KB} = 31,250 \text{ ops/sec}$

- DRAM I/O rate
  - $5.336\text{ GB/sec} / 64\text{KB} = 83,375 \text{ ops/sec}$

- FSB I/O rate
  - Assume we can sustain half the peak rate
  - $5.3 \text{ GB/sec} / 64\text{KB} = 81,540 \text{ ops/sec per FSB}$
  - $163,080 \text{ ops/sec for 2 FSBs}$

- Weakest link: disks
  - $2424 \text{ ops/sec random, 14,000 ops/sec sequential}$
  - Other components have ample headroom to accommodate these rates
Pitfall: Backing Up to Tape

- Magnetic tape used to have advantages
  - Removable, high capacity
- Advantages eroded by disk technology developments
- Makes better sense to replicate data
  - E.g, RAID, remote mirroring
Fallacy: Disk Scheduling

- Best to let the OS schedule disk accesses
  - But modern drives deal with logical block addresses
    - Map to physical track, cylinder, sector locations
    - Also, blocks are cached by the drive
  - OS is unaware of physical locations
    - Reordering can reduce performance
    - Depending on placement and caching
Concluding Remarks

- I/O performance measures
  - Throughput, response time
  - Dependability and cost also important
- Buses used to connect CPU, memory, I/O controllers
  - Polling, interrupts, DMA
- I/O benchmarks
  - TPC, SPECSFS, SPECWeb
- RAID
  - Improves performance and dependability