Hard Disk Drives (HDDs)

Jin-Soo Kim (jinsookim@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu
Three Pieces

- **Virtualization**
  - Virtual CPUs
  - Virtual memory

- **Concurrency**
  - Threads
  - Synchronization

- **Persistence**
  - How to make information persist, despite computer crashes, disk failures, or power outages?
  - Storage
  - File systems
Modern System Architecture

“Broadwell” Intel® Xeon® Processor E5-2600 v4 Product Family

- Up to 22 cores
- 4.8 GHz
- 2 Intel® QPI Links
- 19.2 GB/s per channel

Intel® Xeon® Processor E5-2600 v4 Product Family

- Up to 40 GbE
- 19.2 GB/s per link
- 1 GB/s per lane

Intel® C612 Chipset

- Up to 2 GB/s
- x4 DMI 2.0

Platform Controller Hub (PCH)

- Up to 600 MB/s
- 10 SATA Gen 3 Ports (6 Gb/s)

- Up to 400 MB/s
- 6 USB 3.0 Ports

- 8 USB 2.0 Ports

- 4 Memory Channels DDR4 / Up to 2400 MHz

PCH

- Up to 40 Ports

- PCIe® 3.0

- Up to 1536 GB

- 500 MB/s per lane

- 8 PCIe® 2.0 Ports

Intel® Ethernet Controller XL710 Family

- Up to 40 GbE

- 500 MB/s per lane

- 8 PCIe® 2.0 Ports
A Typical I/O Device

- **Control:** Special instructions (e.g. `in` & `out` in x86) vs. memory-mapped I/O (e.g. `load` & `store`)
- **Data transfer:** Programmed I/O (PIO) vs. DMA
- **Status check:** Polling vs. Interrupts
Classifying I/O Devices

- **Block device**
  - Stores information in fixed-size blocks, each one with its own address
  - Typically, 512B or 4KB per block
  - Can read or write each block independently
  - Disks, tapes, etc.

- **Character device**
  - Delivers or accepts a stream of characters
  - Not addressable and no seek operation supported
  - Printers, networks, mouse, keyboard, etc.
I/O Stack

I/O request

User processes

Device-independent software

Device drivers

Interrupt handlers

Hardware

I/O reply

Make I/O call, format I/O, spooling

Naming, protection, blocking, buffering, allocation

Set up device registers, check status

Wake up driver when I/O completed

Perform I/O operation
Device Drivers

▪ Device-specific code to control each I/O device
  • Require to define a well-defined model and a standard interface

▪ Implementation
  • Statically linked with the kernel
  • Selectively loaded into the system during boot time
  • Dynamically loaded into the system during execution (especially for hot pluggable devices)

▪ Variety is a challenge
  • Many, many devices
  • Each has its own protocol
OS Reliability
OS Reliability and Device Drivers

- Reliability remains a crucial, but unresolved problem
  - 5% of Windows systems crash every day
  - Huge cost of failures: stock exchange, e-commerce, etc.
  - Growing “unmanaged systems”: digital appliances, CE devices

- OS extensions are increasingly prevalent
  - 70% of Linux kernel code
  - Over 35,000 drivers with over 120,000 versions on WinXP
  - Written by less experienced programmer

- Extensions are a leading cause of OS failure
  - Drivers cause 85% of WinXP crashes
  - Drivers are 7 times buggier than the kernel in Linux
Secondary Storage

- Anything that is outside of “primary memory”
  - Does not permit direct execution of instructions or data retrieval via machine load/instructions
  - Abstracted as an array of sectors
  - Each sector is typically 512 bytes or 4096 bytes

- HDD (Hard Disk Drive) Characteristics
  - It’s large: 100 GB or more
  - It’s cheap: 4TB SATA3 hard disk costs 150,000won
  - It’s persistent: data survives power loss
  - It’s slow: milliseconds to access
HDD Architecture

Electromechanical
- Rotating disks
- Arm assembly

Electronics
- Disk controller
- Buffer
- Host interface
A Modern HDD

- Seagate Barracuda ST5000DM000 (5TB)
  - 8 Heads, 4 Discs
  - 63 sectors/track, 16,383 cylinders
  - Avg. track density: 455K TPI (tracks/inch)
  - Avg. areal density: 826 Gbits/sq.inch
  - Spindle speed: 7200 rpm (8.3 ms/rotation)
  - Internal cache buffer: 128 MB
  - Average seek time: < 12.0 ms
  - Max. I/O data transfer rate: 600 MB/s (SATA3)
  - Max. sustained data transfer rate: 160 MB/s
  - Max power-on to ready: < 22.0 sec
HDD Internals

- Our Boeing 747 will fly at the altitude of only a few mm at the speed of approximately 65mph periodically landing and taking off.
- And still the surface of the runway, which consists of a few mm-think layers, will stay intact for years.
Interfacing with HDDs

- **Cylinder-Head-Sector (CHS) scheme**
  - Each block is addressed by <Cylinder #, Head #, Sector #>
  - The OS needs to know all disk “geometry” parameters

- **Logical block addressing (LBA) scheme**
  - First introduced in SCSI
  - Disk is abstracted as a logical array of blocks [0, …, N-1]
  - Address a block with a “logical block address (LBA)”
  - Disk maps an LBA to its physical location
  - Physical parameters of a disk are hidden from OS

```
Read <2, 2>
Write <8, 4>
```

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 ...
```
HDD Performance Factors

- **Seek time** ($T_{\text{seek}}$)
  - Moving the disk arm to the correct cylinder
  - Depends on the cylinder distance (not purely linear cost)
  - Average seek time is roughly one-third of the full seek time

- **Rotational delay** ($T_{\text{rotation}}$)
  - Waiting for the sector to rotate under head
  - Depends on rotations per minute (RPM)
  - 5400, 7200 RPM common, 10K or 15K RPM for servers

- **Transfer time** ($T_{\text{transfer}}$)
  - Transferring data from surface into disk controller, sending it back to the host
# HDD Performance Comparison

<table>
<thead>
<tr>
<th></th>
<th>Cheetah 15K.5</th>
<th>Barracuda</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Capacity</strong></td>
<td>300 GB</td>
<td>1 TB</td>
</tr>
<tr>
<td><strong>RPM</strong></td>
<td>15,000</td>
<td>7,200</td>
</tr>
<tr>
<td><strong>Avg. Seek</strong></td>
<td>4 ms</td>
<td>9 ms</td>
</tr>
<tr>
<td><strong>Max Transfer</strong></td>
<td>125 MB/s</td>
<td>105 MB/s</td>
</tr>
<tr>
<td><strong>Platters</strong></td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td><strong>Cache</strong></td>
<td>16MB</td>
<td>16/32 MB</td>
</tr>
<tr>
<td><strong>Interface</strong></td>
<td>SCSI</td>
<td>SATA</td>
</tr>
<tr>
<td><strong>Random Read</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4 KB)</td>
<td>$T_{\text{seek}} = 4ms$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$T_{\text{rotation}} = 60 / 15000 / 2 = 2ms$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$T_{\text{transfer}} = 4KB / 125MB = 32\mu s$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$R_{I/O} \approx 4KB / 6ms = 0.66$ MB/s</td>
<td></td>
</tr>
<tr>
<td><strong>Sequential Read</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(100 MB)</td>
<td>$T_{\text{transfer}} = 100MB / 125MB = 0.8s$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$R_{I/O} \approx 100MB / 0.8s = 125$ MB/s</td>
<td></td>
</tr>
</tbody>
</table>

$T_{\text{seek}} = 9ms$,  
$T_{\text{rotation}} = 60 / 7200 / 2 = 4.2ms$  
$T_{\text{transfer}} = 4KB / 105MB = 37\mu s$  
$R_{I/O} \approx 4KB / 13.2ms = 0.31$ MB/s  
$T_{\text{transfer}} = 100MB / 105MB = 0.95s$  
$R_{I/O} \approx 100MB / 0.95s = 105$ MB/s
Disk Scheduling

- Given a stream of I/O requests, in what order should they be served?
  - Much different than CPU scheduling
  - Seeks are so expensive
  - Position of disk head relative to request position matters more than length of a job

- Work conserving schedulers
  - Always try to do work if there’s work to be done

- Non-work-conserving schedulers
  - Sometimes, it’s better to wait instead if system anticipates another request will arrive
FCFS

- **First-Come First-Served (\(=\) do nothing)**
  - Reasonable when load is low
  - Long waiting times for long request queues
SSTF

- Shortest Seek Time First
  - Minimizes arm movement (seek time)
  - Unfairly favors middle blocks
  - May cause starvation

- Nearest-Block-First (NBF) when the drive geometry is not available to the host OS
SCAN

- SCAN
  - Service requests in one direction until done, then reverse
  - Skews wait times non-uniformly
  - Favors middle blocks

- F-SCAN
  - Freezes the queue when it is doing a sweep
  - Avoids starvation of far-away requests
C-SCAN

- **Circular SCAN**
  - Like SCAN, but only goes in one direction (e.g. typewriter)
  - Uniform wait times

- **SCAN and C-SCAN** are referred to as the “elevator” algorithm
  - Both do not consider rotation
Modern Disk Scheduling

- **I/O scheduler in the host OS**
  - Improve overall disk throughput
    - Merge requests to reduce the number of requests
    - Sort requests to reduce disk seek time
  - Prevent starvation
  - Provide fairness among different processes

- **Disk drive**
  - Disk has multiple outstanding requests
    - e.g. SATA NCQ (Native Command Queueing): up to 32 requests
  - Disk schedules requests using its knowledge of head position and track layout
    - e.g. SPTF (Shortest Positioning Time First): consider rotation as well