

Operating Systems

CSCI 3150

Lecture 13: File Systems

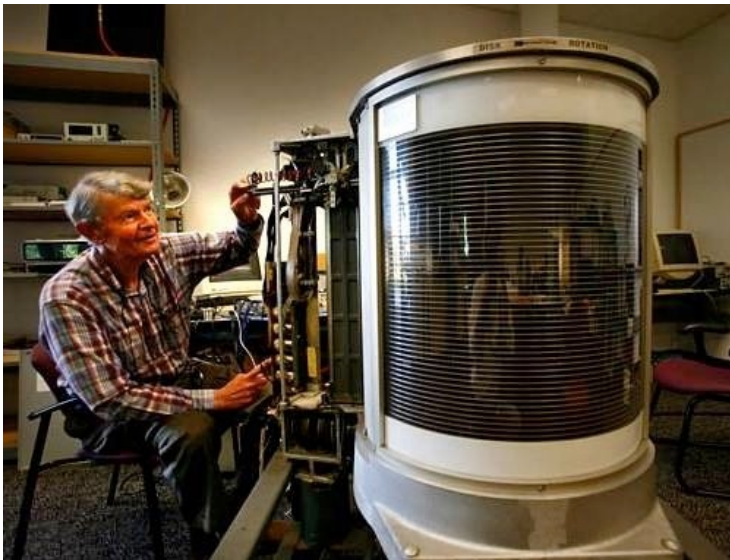
Part I: I/O Devices, HDD

Hong Xu

<https://github.com/henryhxu/CSCI3150>

What problems are we solving?

- ▣ **Persistent** data storage and access
- ▣ Fast-growing industry

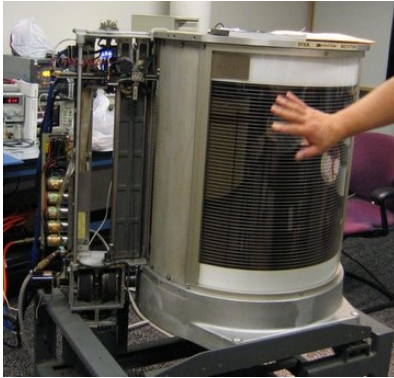


1956, IBM, 24 inches,
3.75MB, 1KB/sec, >\$150k

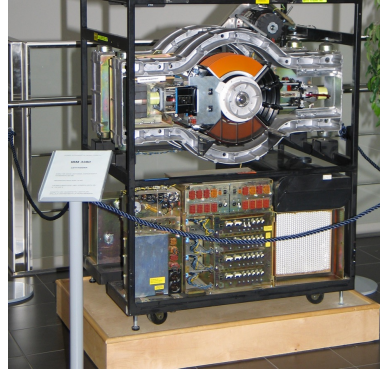


2015, Seagate, 3.5 inches,
5TB, 6GB/sec, <\$200

History of storage technology



1956: IBM 350, 24 inches, 3.75MB, 1KB/sec, > \$150,000



1980: IBM 3380, first GB disk (1.26G), > \$100K



1980: ST-560, first 5.25 inch drive, 5MB, \$1500



Tape (DECtape): primary storage for main-frames and mini-computers (1950 ~ 70s)



NextCom notebook 2006: first laptop w/ SSD as storage

History is heavy...



Source: <https://en.wikipedia.org/wiki/File:SixHardDriveFormFactors.jpg/>

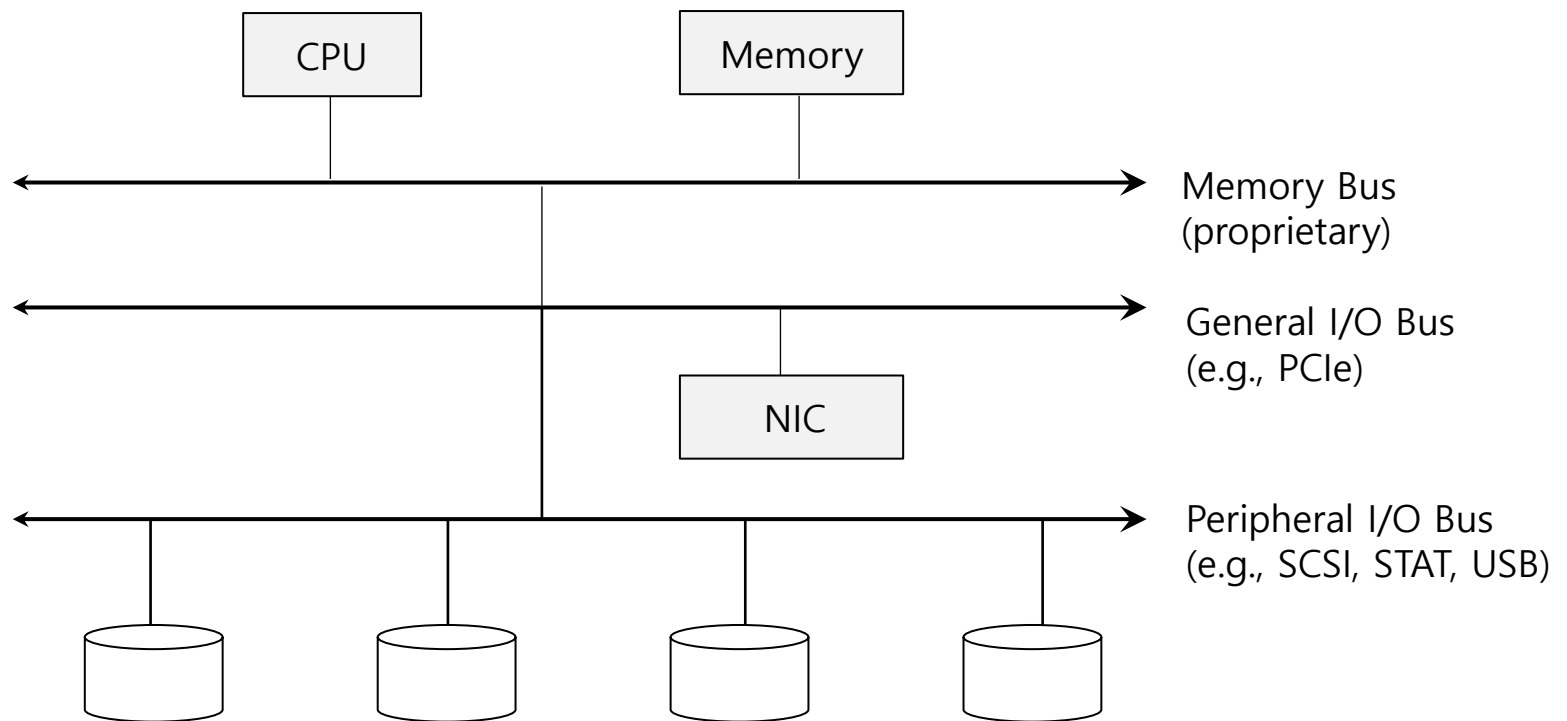
File systems: Agenda

- ▣ First we will discuss “generic I/O devices and abstractions”...
- ▣ ... and properties of physical hard disks
- ▣ Then we will discuss how we build file systems on them
 - ◆ Files, directories
 - ◆ Sharing, protection
 - ◆ File system layout, design
 - ◆ ...
- ▣ Lastly, we will cover SSD

I/O Devices

- ▣ Input/Output, I/O, is **critical** to **interact** with computer systems
- ▣ Issues:
 - ◆ How should I/O be integrated into systems?
 - ◆ What are the general mechanisms?
 - ◆ How can we make it efficiently?

Structure I/O Devices



Prototypical System Architecture

CPU is attached to the main memory of the system via some kind of memory **bus.**
Some devices are connected to the system via a **I/O bus.**

I/O Architecture

- ▣ Buses

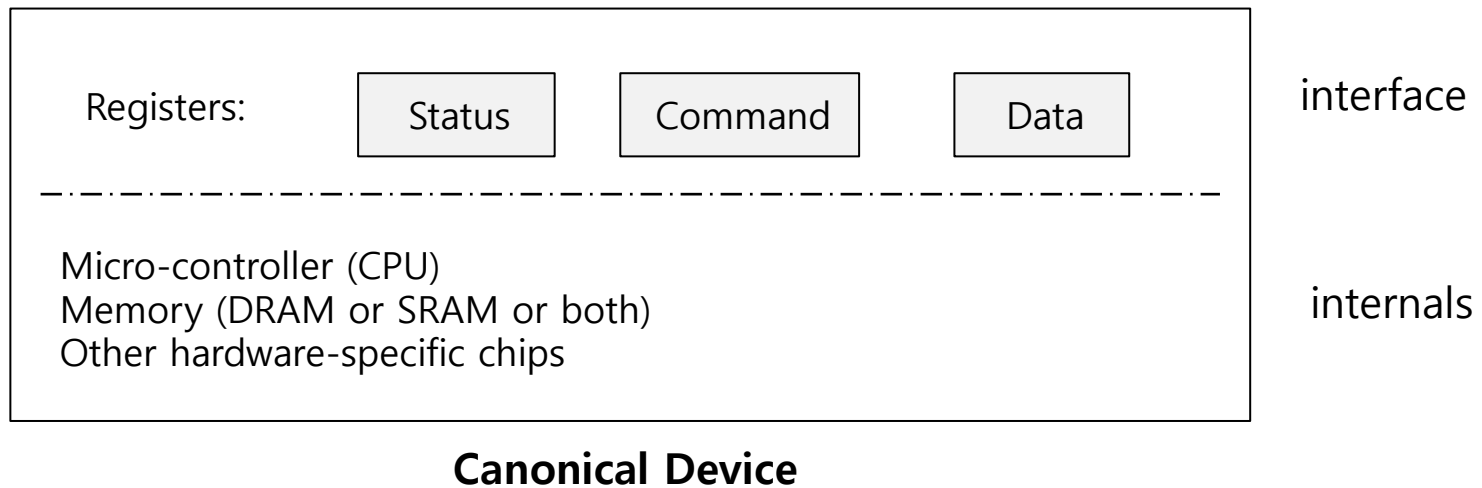
- ◆ Data paths to enable information exchange between CPU, RAM, and I/O devices.

- ▣ I/O bus

- ◆ Data path that connects a CPU to an I/O device.
- ◆ I/O bus is connected to I/O devices by three hardware components: I/O ports, interfaces and device controllers.

Canonical Device

- ▣ A Canonical Device has two important components.
 - ◆ **Hardware interface** allows the system software to control its operation.
 - ◆ **Internals** which are implementation specific.



Hardware interface of the canonical device

- ▣ **status register**

- ◆ See the current status of the device

- ▣ **command register**

- ◆ Tell the device to perform a certain task

- ▣ **data register**

- ◆ Pass data to the device, or get data from the device

By reading and writing above **three registers,
the operating system can **control device behavior**.**

Hardware interface of the canonical device (Cont.)

▣ Typical interaction

```
while (STATUS == BUSY)
    ; //wait until device is not busy
write data to data register
write command to command register
    (doing so starts the device and executes the command)
while (STATUS == BUSY)
    ; //wait until device is done with your request
```

Polling

- ❑ Operating system waits until the device is ready by **repeatedly** reading the status register.
 - ◆ Simple; responsive
 - ◆ **However, it wastes (lots of!!) CPU time just waiting for the device**
 - Switching to another ready process is better in terms of utilizing the CPU.

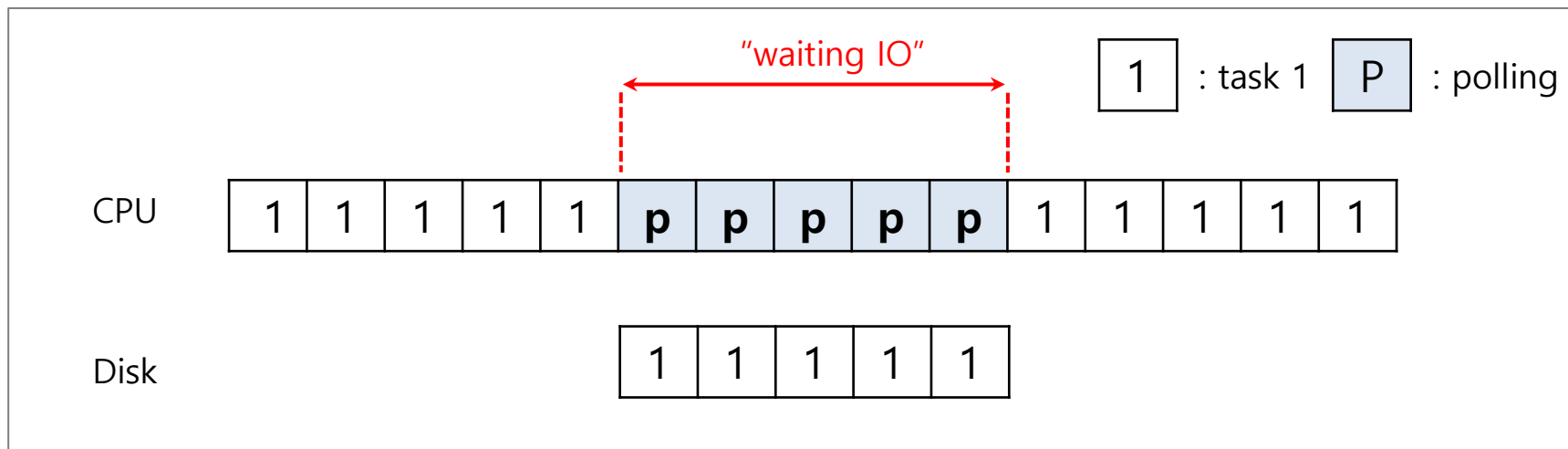


Diagram of CPU utilization with polling

Interrupts

- Put the process waiting on I/O to sleep and context-switch to another
- When the I/O finishes, wake up the sleeping process by **interrupts**
 - ◆ Enable **CPU and the disk to be better utilized**

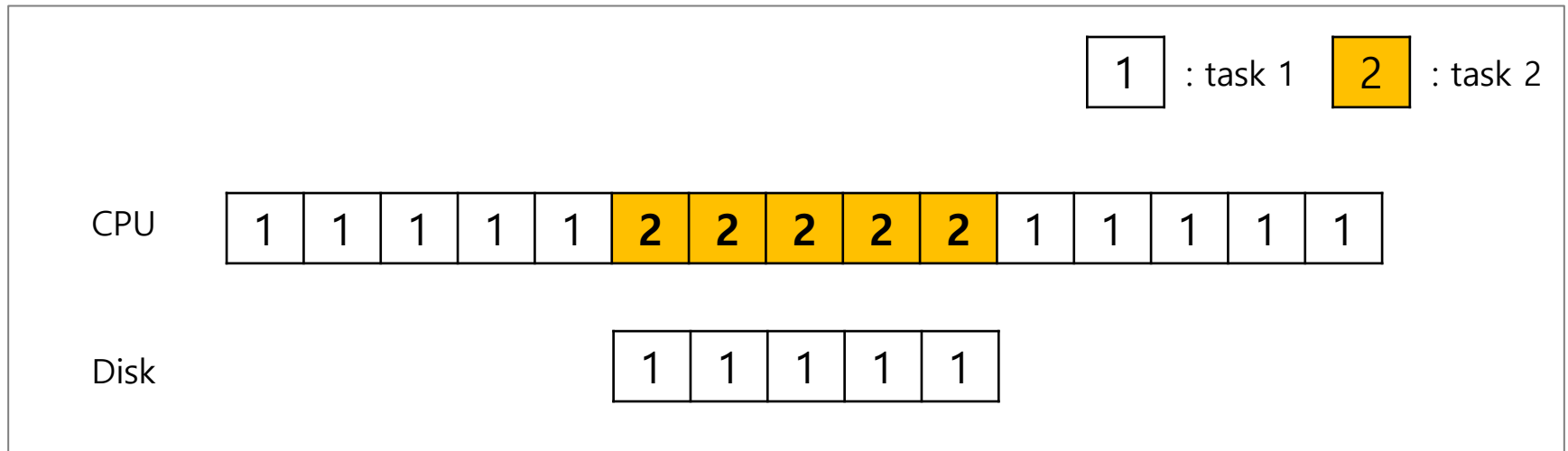


Diagram of CPU utilization with interrupts

Polling vs interrupts

❑ Interrupt is not always the best solution

- ◆ If, device performs very quickly, interrupts will “slow down” the system
- ◆ Because **context switch is expensive (switching to another process)**

If a device is fast → **polling** is better.
If it is slow → **interrupt** is better.

CPU is once again over-burdened

- ❑ CPU **wastes a lot of time** to copy *a large chunk of data* from memory to the device.

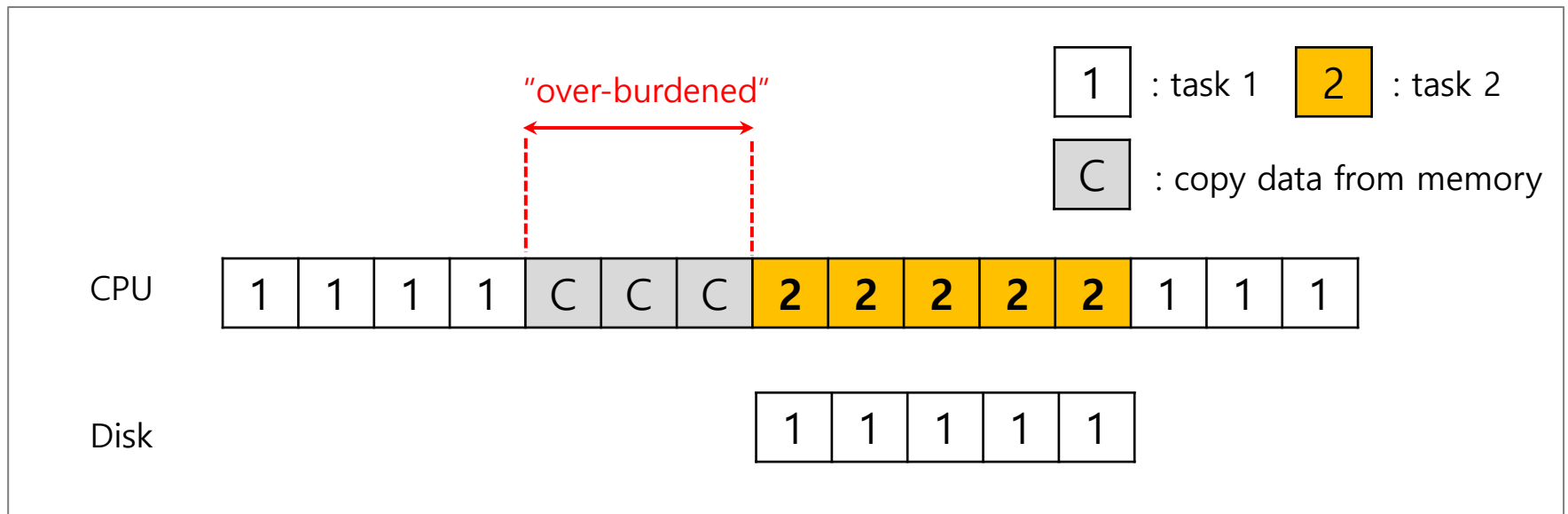


Diagram of CPU utilization

DMA (Direct Memory Access)

- **Copy data** in memory by knowing “where the data lives in memory, how much data to copy”
- When completed, DMA raises an interrupt, I/O begins on disk

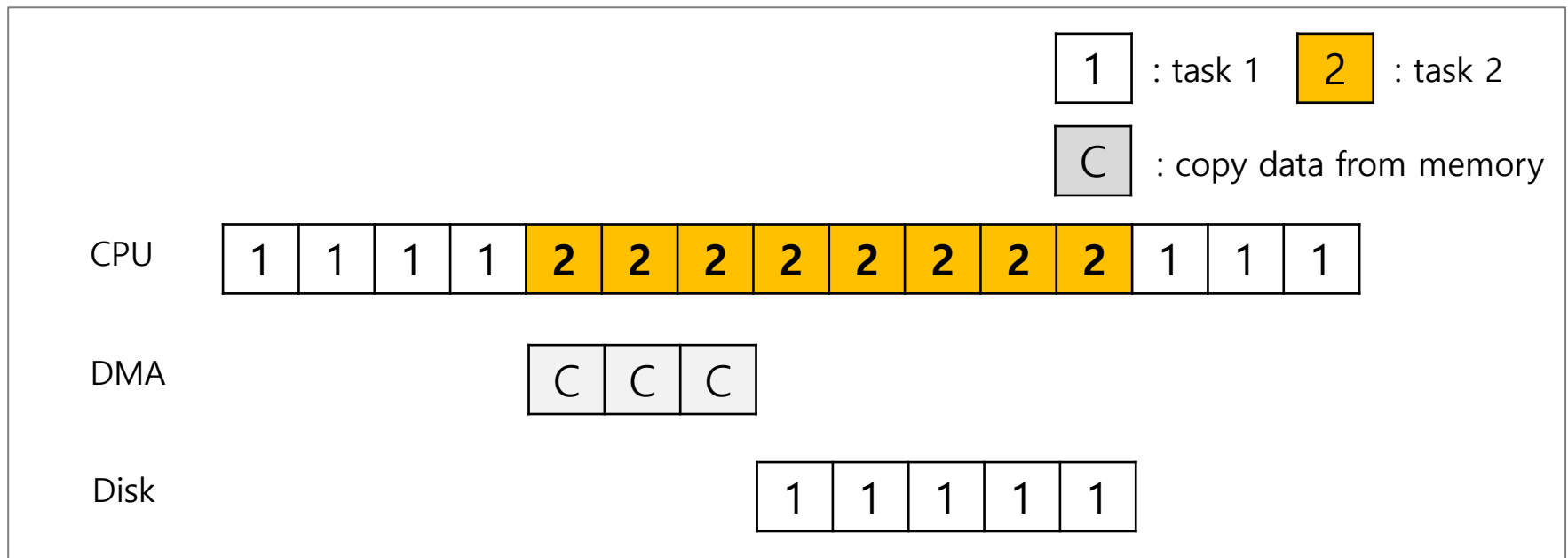


Diagram of CPU utilization by DMA

Device interaction

- ▣ How the OS communicates with the **device**?
- ▣ Solutions
 - ◆ **I/O instructions**: a way for the OS to send data to specific device registers.
 - `in` and `out` instructions on x86
 - ◆ **memory-mapped I/O**
 - Device registers available as if they were memory locations.
 - The OS `load` (to read) or `store` (to write) to the device instead of main memory.

Device interaction (Cont.)

- How the OS interact with **different types of interfaces**?
 - ◆ Ex) We'd like to build a file system that worked on top of SCSI disks, IDE disks, USB keychain drivers, and so on.
- Solutions: **Abstraction**
 - ◆ Encapsulate **specifics of device interaction**

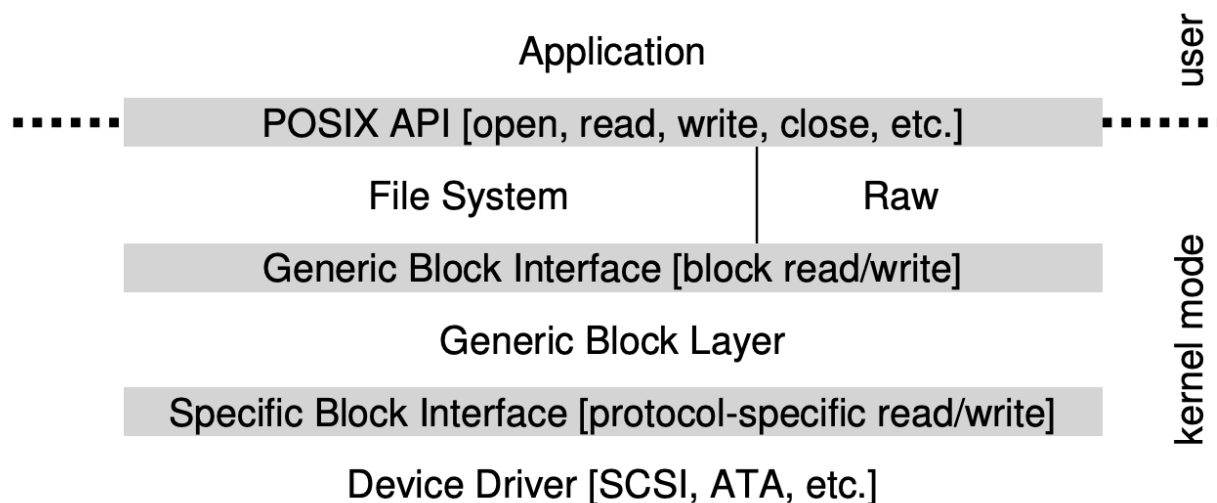


Figure 36.4: The File System Stack

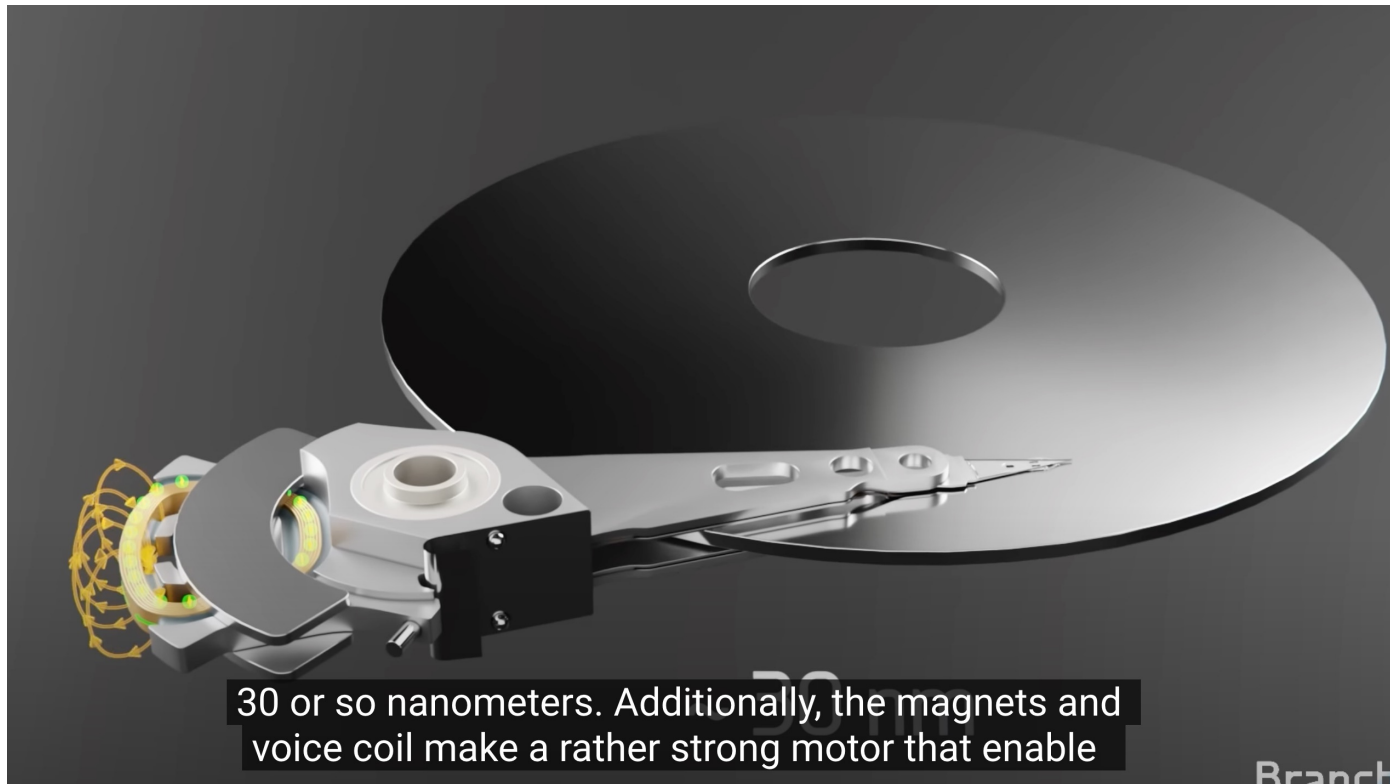
Summary

- ▣ To save the CPU cycles for IO
 - ◆ Interrupt
 - ◆ DMA
- ▣ To access the device registers
 - ◆ Memory-mapped IO
 - ◆ Explicit IO instructions

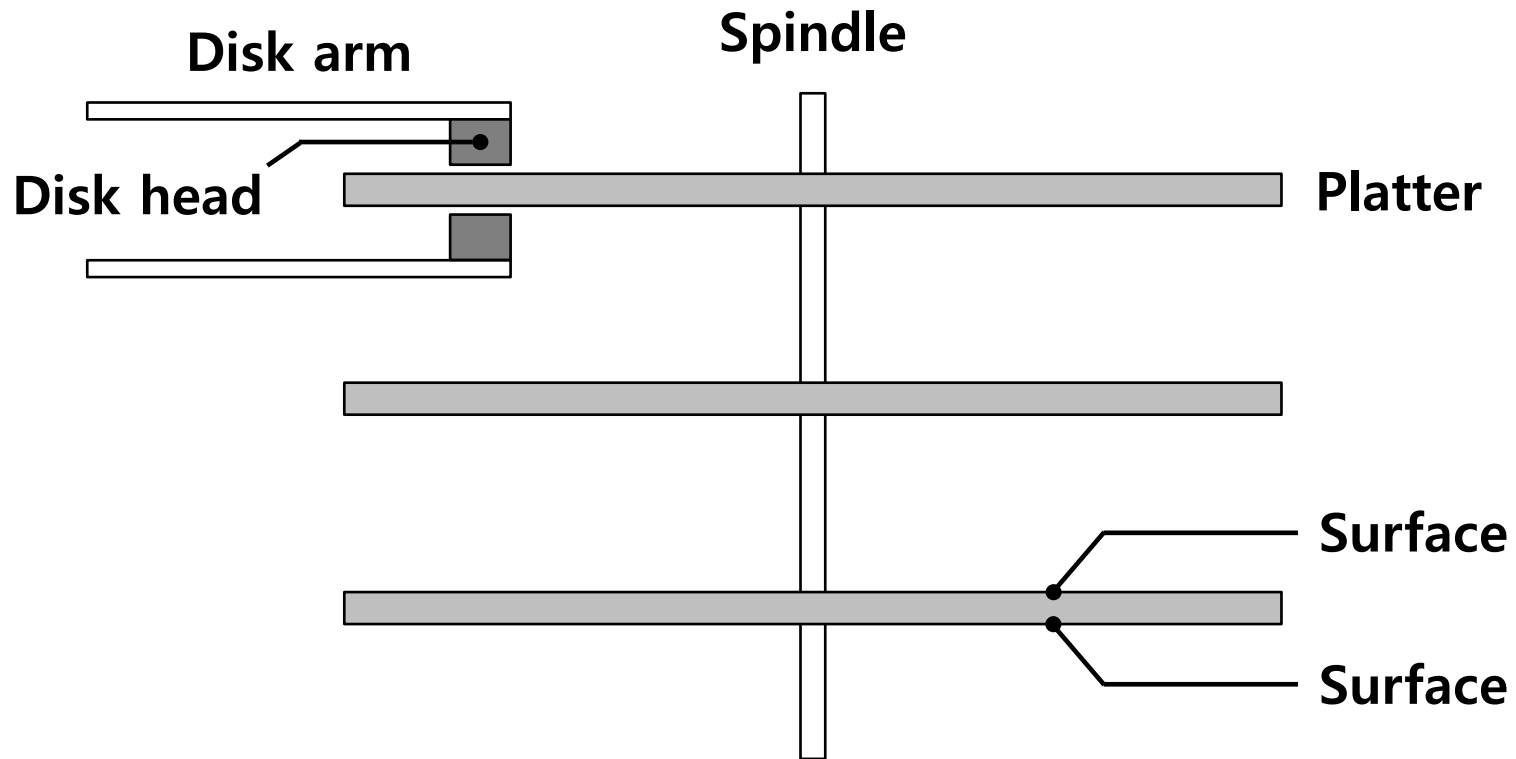
Hard Disk Drive (HDD)

How does it work?

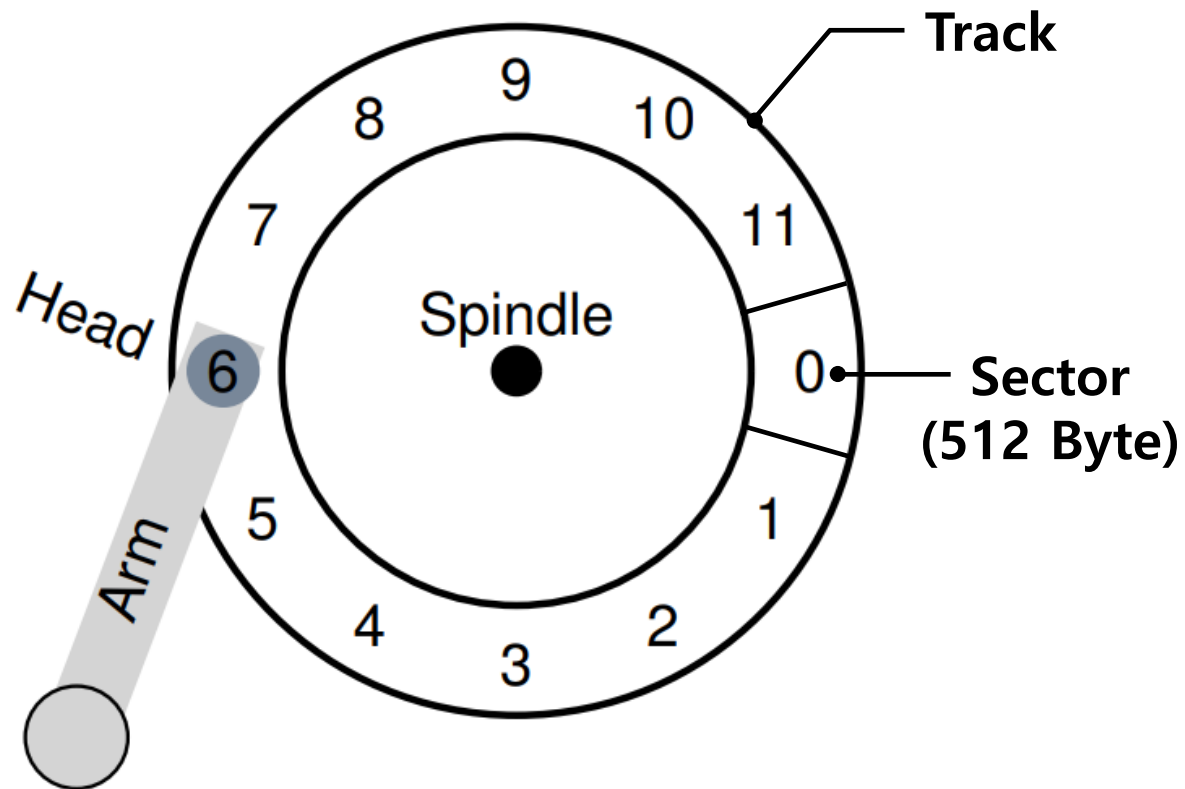
- ❑ Physics, electro-magnetic field
- ❑ <https://www.youtube.com/watch?v=wteUW2sL7bc> (physics)
- ❑ <https://www.youtube.com/watch?v=wtdnatmVdlg> (components)



Basic Geometry

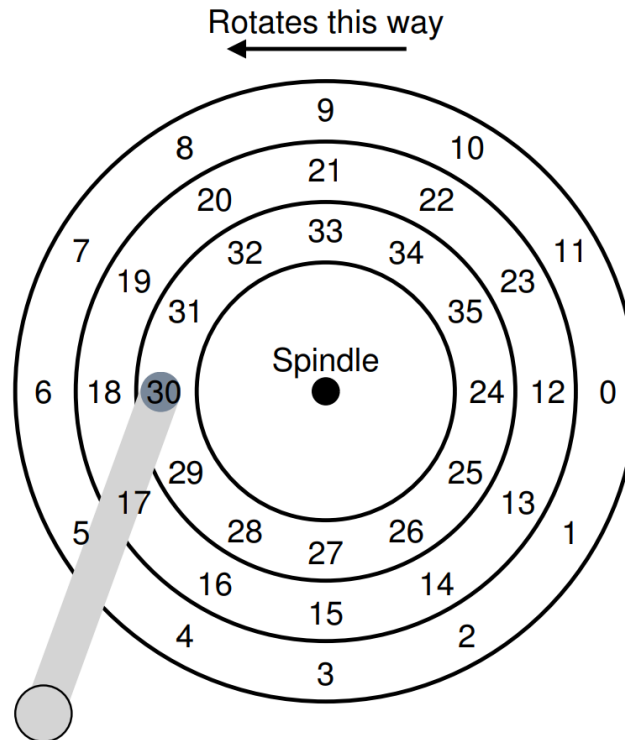


Basic Geometry (Cont.)



A Simple Disk Drive

▣ Rotation Delay



RPM, rotations per minute

- 7200 to 15000 RPM
- 10000 RPM -> ~6ms per rotation

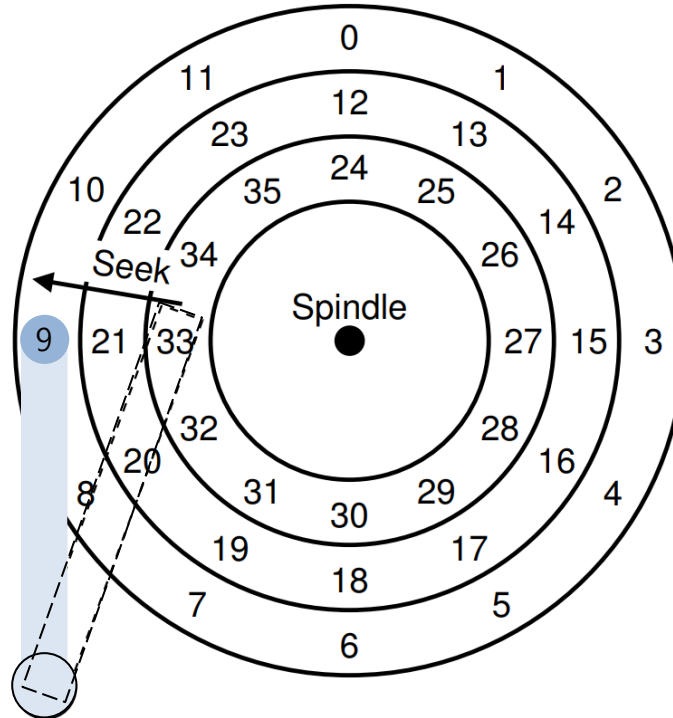
If the full rotation delay is R ,

the rotation delay of $(30 \rightarrow 24)$ is $\frac{R}{2}$

A Simple Disk Drive (Cont.)

▣ Seek Time

- ◆ To read sector 11

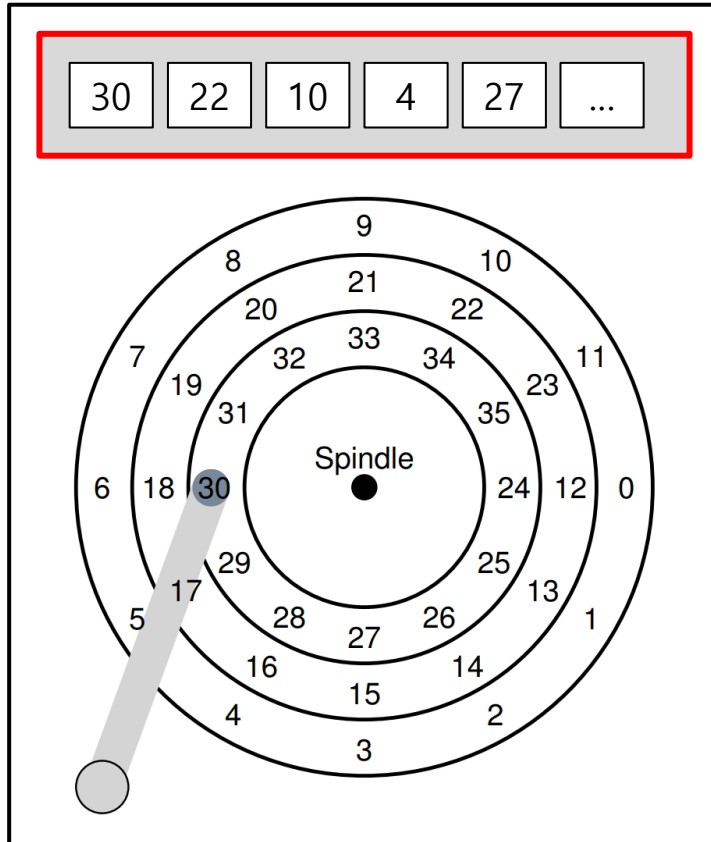


Phases of seek

Acceleration → *Coasting* → *Deceleration* → *Settling time*
(about 0.5~2 ms)

A Simple Disk Drive (Cont.)

Cache (Track buffer)



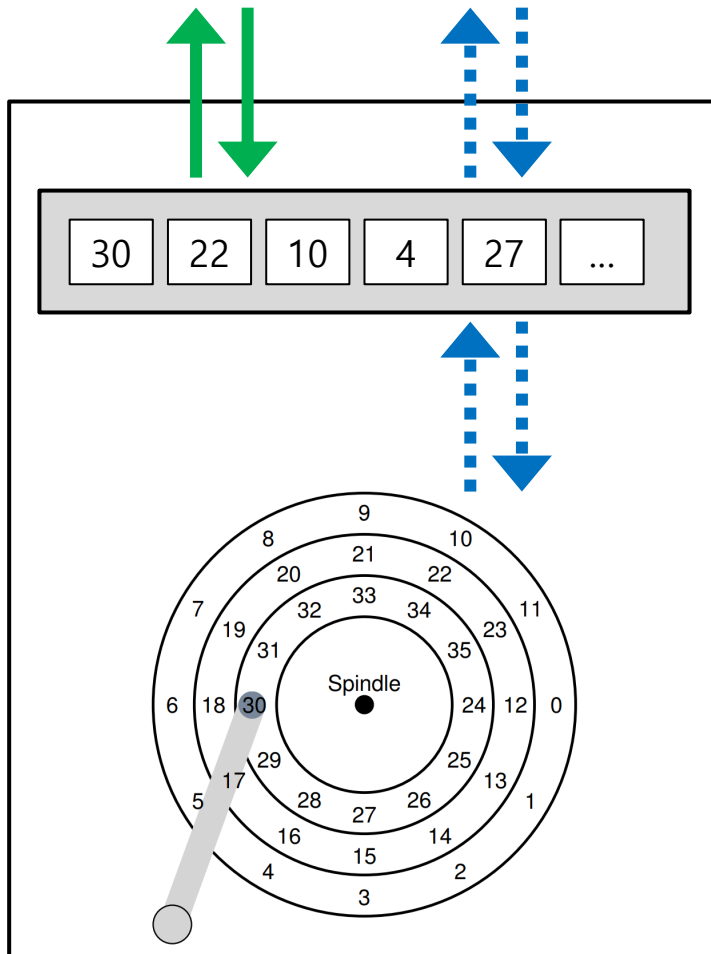
Small amount of memory
(usually around 8 or 16MB)

Hold data read from or written to the disk

Allow the drive to quickly respond to requests

A Simple Disk Drive (Cont.)

Cache (Track buffer)



→ Write-Back

Acknowledge the write has completed when it has put the data in its memory

→ Write-Through

Acknowledge after the write has actually been written to disk

I/O Time: Doing The Math

- ▣ I/O Time

$$T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$$

- ▣ I/O Rate

$$R_{I/O} = \frac{Size_{Transfer}}{T_{I/O}}$$

I/O Time: Doing The Math (Cont.)

▣ 4KB Random Write Example

	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

$$T_{seek} = 4\text{ms}$$

$$T_{rotation} = 15,000 \text{ RPM} (= 250\text{RPS} = 4\text{ms} / 1 \text{ rotation}) / 2 \\ = 2\text{ms}$$

$$T_{transfer} = 4\text{KB} / 125(\text{MB/s}) \\ = 30\mu\text{s}$$

$$T_{I/O} = 4\text{ms} + 2\text{ms} + 30\mu\text{s} \approx 6\text{ms}$$

$$R_{I/O} = 4\text{KB} / 6\text{ms} = 0.66\text{MB/s}$$

I/O Time: Doing The Math (Cont.)

▣ 4KB Random Write Example (Cont.)

	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

$$T_{seek} = 9\text{ms}$$

$$T_{rotation} = 7,200 \text{ RPM}(= 120\text{RPS} = 8\text{ms} / 1 \text{ rotation}) / 2 \\ = 4\text{ms}$$

$$T_{transfer} = 4\text{KB} / 105(\text{MB/s}) \\ = 38\mu\text{s}$$

$$T_{I/O} = 9\text{ms} + 4\text{ms} + 38\mu\text{s} \approx 13\text{ms}$$

$$R_{I/O} = 4\text{KB} / 13\text{ms} = 0.31\text{MB/s}$$

I/O Time: Doing The Math (Cont.)

▣ Sequential Write Example

	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

$$T_{seek} = 4\text{ms}$$

$$\begin{aligned} T_{rotation} &= 15,000 \text{ RPM} (= 250\text{RPS} = 4\text{ms} / 1 \text{ rotation}) / 2 \\ &= 2\text{ms} \end{aligned}$$

$$\begin{aligned} T_{transfer} &= 100\text{MB} / 125(\text{MB/s}) \\ &= 800\text{ms} \end{aligned}$$

$$T_{I/O} = 4\text{ms} + 2\text{ms} + 800\text{ms} = 806\text{ms} \approx 800\text{ms}$$

$$R_{I/O} = 100\text{MB} / 800\text{ms} = 125\text{MB/s}$$

I/O Time: Doing The Math (Cont.)

▣ Sequential Write Example (Cont.)

	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

$$T_{seek} = 9\text{ms}$$

$$T_{rotation} = 7,200 \text{ RPM}(= 120\text{RPS} = 8\text{ms} / 1 \text{ rotation}) / 2 \\ = 4\text{ms}$$

$$T_{transfer} = 100\text{MB} / 105(\text{MB/s}) \\ = 950\text{ms}$$

$$T_{I/O} = 9\text{ms} + 4\text{ms} + 950\text{ms} = 963\text{ms} \approx 950\text{ms}$$

$$R_{I/O} = 100\text{MB} / 950\text{ms} = 105\text{MB/s}$$

I/O Time: Doing The Math (Cont.)

	Cheetah	Barracuda
$R_{I/O}$ Random	0.66 MB/s	0.31 MB/s
$R_{I/O}$ Sequential	125 MB/s	105 MB/s

Performance vs Capacity

	Cheetah	Barracuda
$R_{I/O}$ Random	0.66 MB/s	0.31 MB/s
$R_{I/O}$ Sequential	125 MB/s	105 MB/s

Random Write vs Sequential Write