Lecture 17: I/O Devices and Hard Disk Internals

Operating Systems

Content taken from: https://pages.cs.wisc.edu/~remzi/OSTEP/

https://www.cse.iitb.ac.in/~mythili/os/

Till now

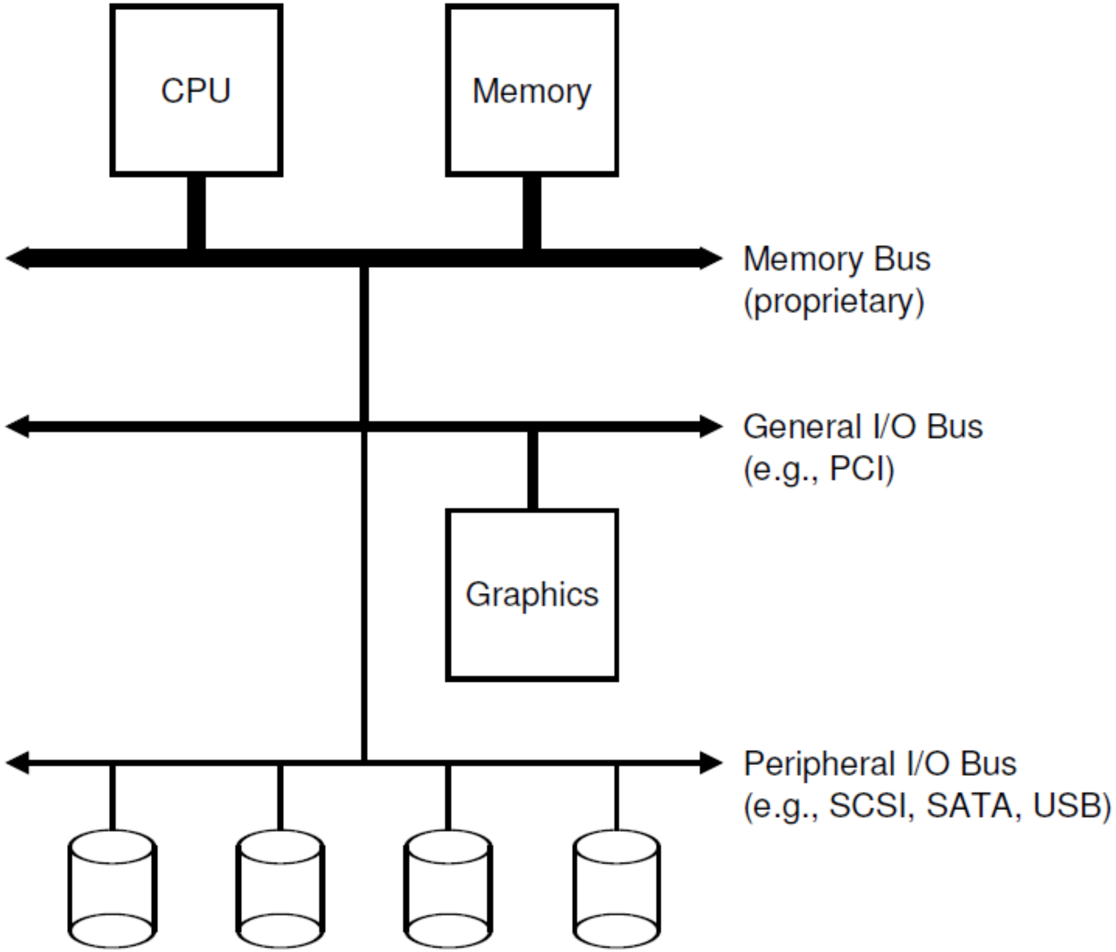
- CPU Virtualization
  - Process abstraction and execution
  - CPU scheduling policies
- Memory virtualization
  - Segmentation and Paging
  - TLB
  - Swapping
- Concurrency
  - Threads
  - Locks
  - Condition variables
  - Semaphores
  - Concurrency bugs
    - Deadlocks

From Today

- We will talk about persistence
- Beginning with I/O devices in general and then shift our focus to storage (hard disk) and file system

I/O Devices

- Input and Output Devices
- I/O devices connect to the CPU and memory via a bus
  - High speed bus, e.g., PCI
  - Other: SCSI, USB, SATA
- Point of connection to the system: port



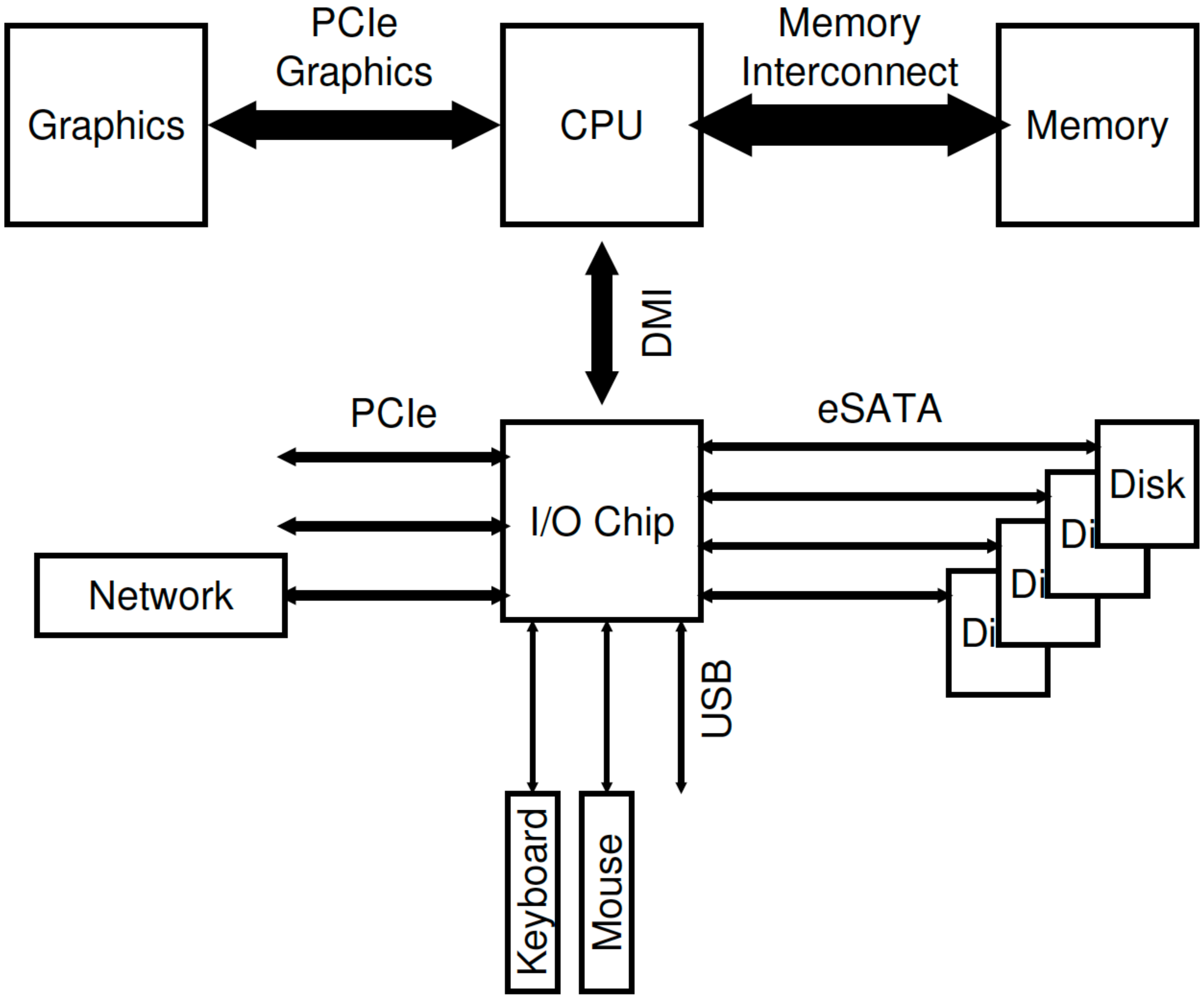


Figure 36.2: **Modern System Architecture**

Simple Device Model

- **Interface:** Devices expose an interface of memory registers
  - Current status of device
  - Command to execute
  - Data to transfer
- **Internals:** The internals of device are usually hidden

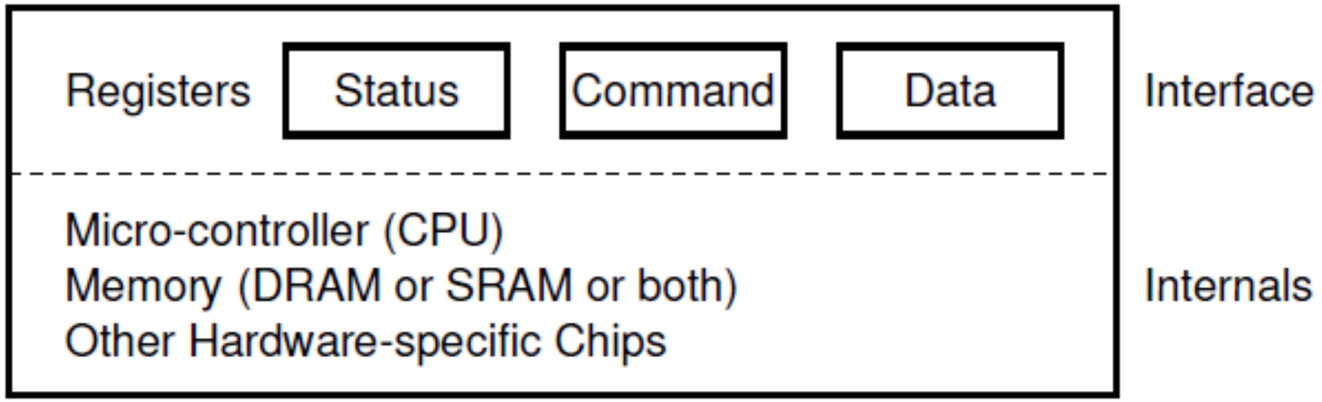


Figure 36.2: **A Canonical Device**

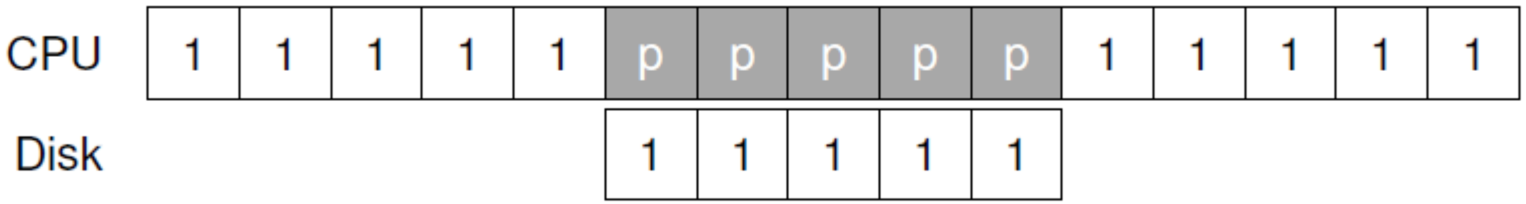
Canonical I/O Protocol

- OS polls the I/O device
- OS writes Data to the device DATA register
- OS tell the device to start execution
- OS waits for the device to finish by polling the device
- Challenges?

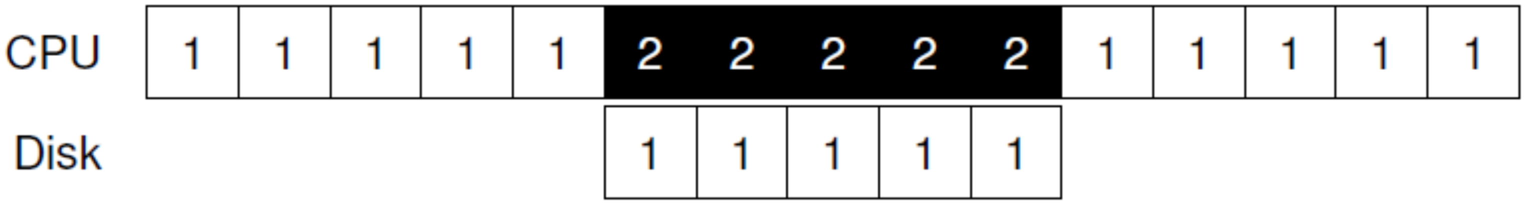
```
While (STATUS == BUSY)
    ; // wait until device is not busy
Write data to DATA register
Write command to COMMAND register
    (Doing so starts the device and executes the command)
While (STATUS == BUSY)
    ; // wait until device is done with your request
```

Lowering CPU Overhead with Interrupts

- Polling wastes CPU cycles



- Instead, OS can put process to sleep and switch to another process



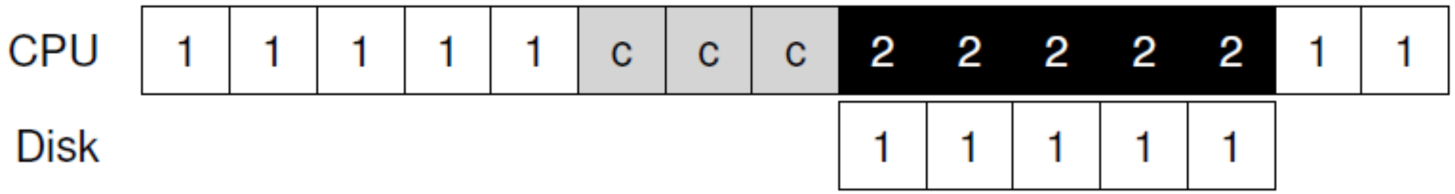
- When I/O request completes, device raises interrupt

Interrupt Handler

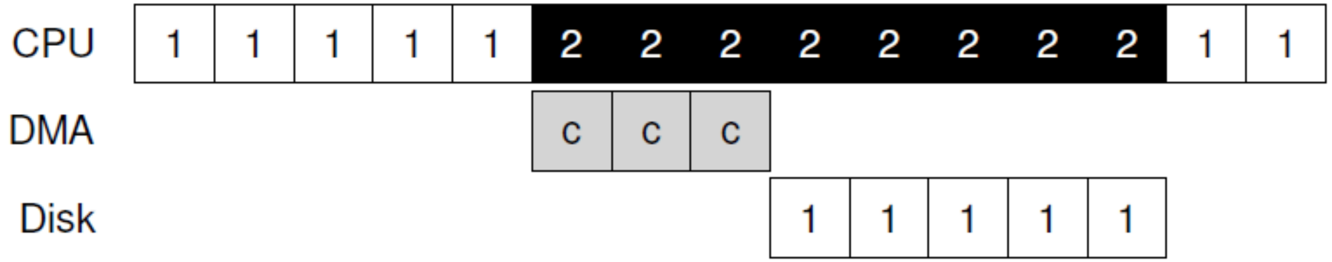
- Interrupt switches process to **kernel** mode
- Interrupt Descriptor Table (IDT) stores pointers to interrupt handlers (interrupt service routines)
  - Interrupt (IRQ) number identifies the interrupt handler to run for a device
- Interrupt handler acts upon device notification, unblocks the process waiting for I/O (if any), and starts next I/O request (if any pending)
- Handling interrupts imposes kernel mode transition overheads
  - Note: polling may be faster than interrupts if device is fast

More Efficient Data Movement with DMA

- CPU cycles wasted in copying data to/from device



- Instead, a special piece of hardware (DMA engine) copies from main memory to device
    - CPU gives DMA engine the memory location of data
    - In case of read, interrupt raised after DMA completes
    - In case of write, disk starts writing after DMA completes

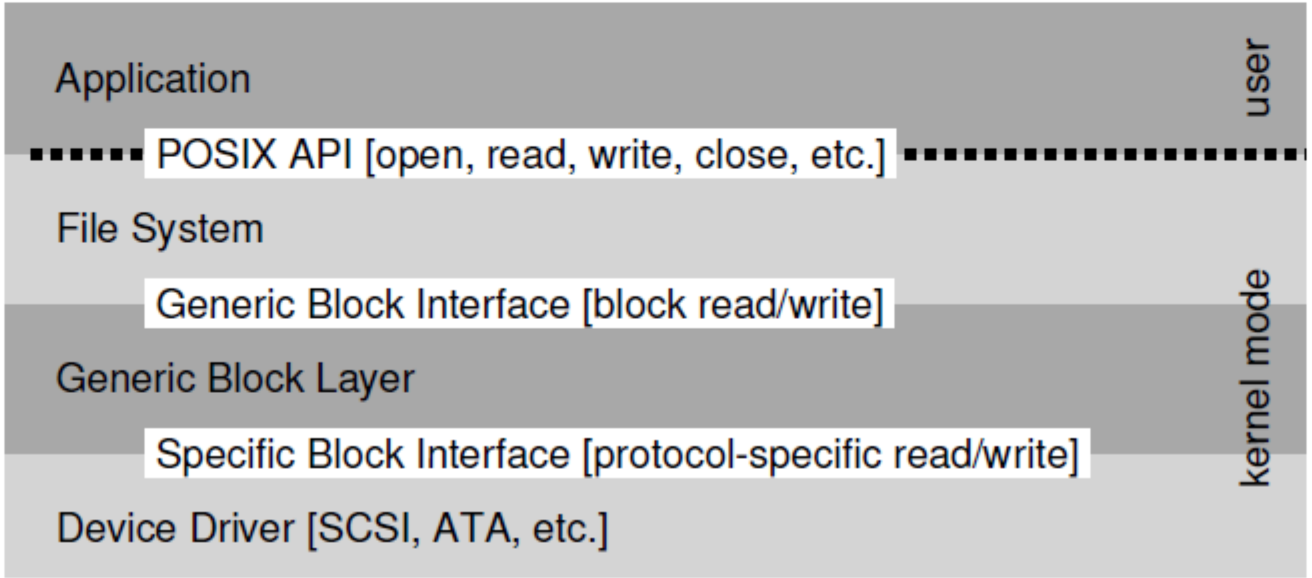


How does OS interact with I/O devices?

- How does OS read/write to registers like status and command?
- Explicit I/O instructions
  - E.g., on x86, `in` and `out` instructions can be used to read and write to specific registers on a device
  - Privileged instructions accessed by OS
- Memory mapped I/O
  - Hardware makes device registers appear like memory locations
  - OS simply reads and writes from memory
  - Memory hardware routes accesses to these special memory addresses to device registers

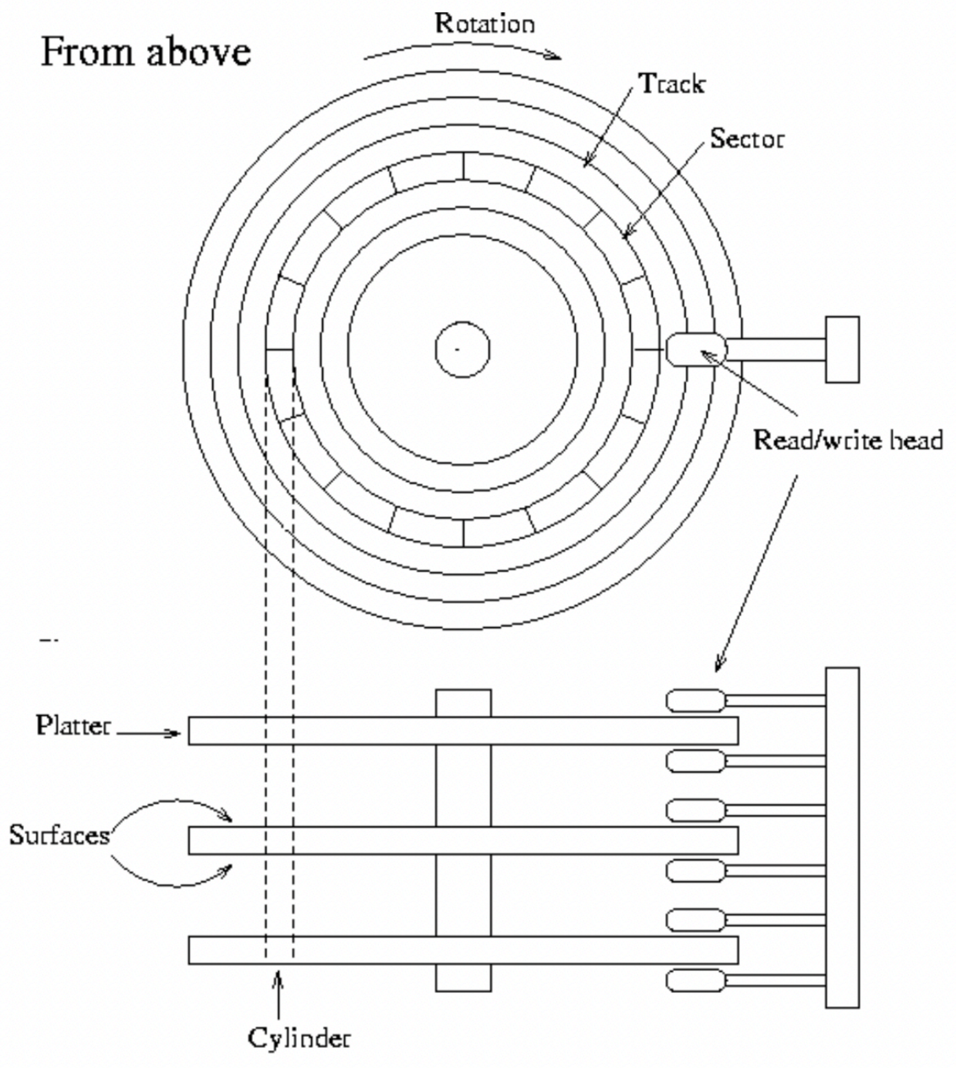
Device Driver

- Device driver: part of OS code that talks to specific device, gives commands, handles interrupts etc.
- Most OS code abstracts the device details
  - E.g., file system code is written on top of a generic block interface



Hard Disk

- Interface: a set of 512-byte blocks (sectors), that can be read or written atomically
  - Sectors are numbered from 0 to N-1
- Internals: one or more circular platters, connected by a spindle, spinning at ~10K RPM (rotations per minute)
- Each platter has a disk head and arm
- A platter is divided into multiple tracks, and each track into 512-byte sectors

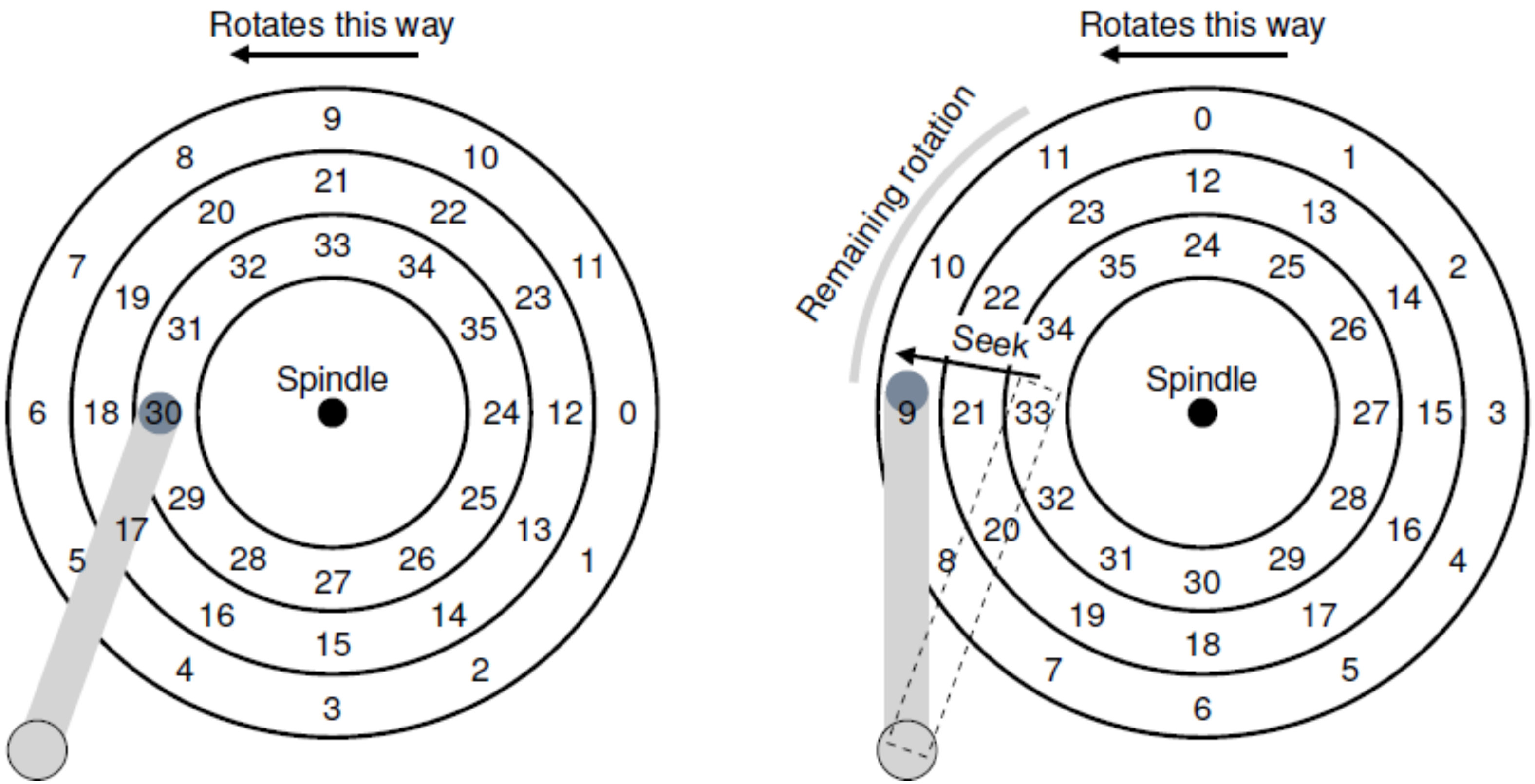


Top and Side View of Hard-disk

https://www.researchgate.net/figure/Top-and-Side-View-of-Hard-disk_fig1_334274392

Accessing a particular sector

- Suppose disk head at 30, need to access 11
- Seek to the correct track, wait for disk to rotate



Time Taken for I/O operation

- Time taken to read/write a block consists of
  - Seek time to get to the right track (few ms)
  - Rotational latency for disk to spin to correct sector on the track (few ms)
  - Transfer time to read sector (few tens microsec)

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

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

Let us solve a simple numerical

- Compute the rate of data transfer for each of the given disks for the following workloads:
  - **Random workload:** Issues small (**4 KB**) reads to random locations on the disk
  - **Sequential workload:** Reads **100 MB** of data in sequence (reads a large number of sectors consecutively from the disk without jumping around)

	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

Figure 37.5: **Disk Drive Specs: SCSI Versus SATA**

- **Cheetah data transfer rate for random access**
  - T(I/O) = T(seek) + T(rotation) + T(transfer)
  - T(seek) = 4 ms (from the table)

  - T(rotation) =
    - 15000 RPM = 250 RPS = 1/250 = 0.004 sec = 4 ms
  - On average, the disk will encounter a half rotation
  - T(rotation) = average rotation time = 2 ms

  - T(transfer) = Size of the transfer over the peak transfer rate
  - Size of the transfer = 4KB
  - Peak transfer rate = 125MB/s
  - T(transfer) = 4KB/125MB/s = 4*1000/125*1000*1000
  - T(transfer) = 0.000032 = 32 micro s = 30 micro s (approx)

  - T(I/O) = 4 ms + 2 ms + 30 micro s = 6 ms (approx)
  - R(I/O) = Size of the transfer/average IO time
  - R(I/O) = 4KB/6ms = 4*1000/6/1000 = 0.66 MB/s

  - **Barracuda data transfer rate for random access**
    - T(seek) = 9 ms (from the table)
    - T(rotation) =
      - 7200 RPM = 120 RPS = 1/120 = 0.008 sec = 8 ms
    - T(rotation) = 4 ms
    - T(transfer) = 4KB/105MB/s = 4*1000/105*1000*1000
    - T(transfer) = 0.000038 = 38 micro secs
    - T(I/O) = 9 ms + 4 ms + 38 micro s = 13 ms (approx)
    - R(I/O) = 4KB/13ms = 4*1000/13/1000 = 0.30 MB/s (approx)

	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

Figure 37.5: **Disk Drive Specs: SCSI Versus SATA**

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

Figure 37.6: **Disk Drive Performance: SCSI Versus SATA**

- **Cheetah data transfer rate for sequential access**
  - T(I/O) = T(seek) + T(rotation) + T(transfer)
  - T(seek) = 4 ms (from the table)
  - T(rotation) =
    - 15000 RPM = 250 RPS = 1/250 = 0.004 sec = 4 ms
  - On average, the disk will encounter a half rotation
  - T(rotation) = average rotation time = 2 ms
  - T(transfer) = Size of the transfer over the peak transfer rate
  - Size of the transfer = 100MB
  - Peak transfer rate = 125MB/s
  - T(transfer) = 100MB/125MB/s = 0.8 s
  - T(I/O) = 4 ms + 2 ms + 0.8 s = 0.806 s = 800 ms (approx)
  - R(I/O) = Size of the transfer/average IO time
  - R(I/O) = 100MB/800ms = 125 MB/s
  - **Barracuda data transfer rate for sequential access**
    - T(seek) = 9 ms (from the table)
    - T(rotation) =
      - 7200 RPM = 120 RPS = 1/120 = 0.008 sec = 8 ms
    - T(rotation) = 4 ms
    - T(transfer) = 100MB/105MB/s = 0.95 s
    - T(I/O) = 9 ms + 4 ms + 0.95 s = 950 ms (approx)
    - R(I/O) = 100MB/950ms = 105 MB/s (approx)

	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

Figure 37.5: **Disk Drive Specs: SCSI Versus SATA**

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

Figure 37.6: **Disk Drive Performance: SCSI Versus SATA**