# 37. Hard Disk Drives

Operating Systems in Three Easy Pieces

**Keith Perkins**

Original slides by
Youjip Won
Hanyang University

Embedded Software Systems Lab

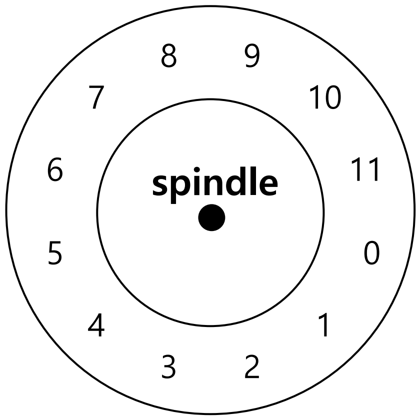
## Hard Disk Drives

- Hard disk drives have been **the main form of persistent data storage** in computer systems for decades.
  - The drive consists of a large number of **sectors** (a 512-byte block).
  - We can view the disk with `n` sectors as an array of sectors; `0` to `n-1`.

## Interface

- The only guarantee is that a single 512-byte write is **atomic**.
- Multi-sector operations are possible.
  - Many file systems will read or write 4KB at a time.
  - **Torn write**:
    - If an untimely power loss occurs, only a portion of a larger write may complete.
- Accessing blocks in **a contiguous chunk** is the fastest access mode.
  - A sequential read or write
  - Much faster than any more random access pattern.

## Basic Geometry



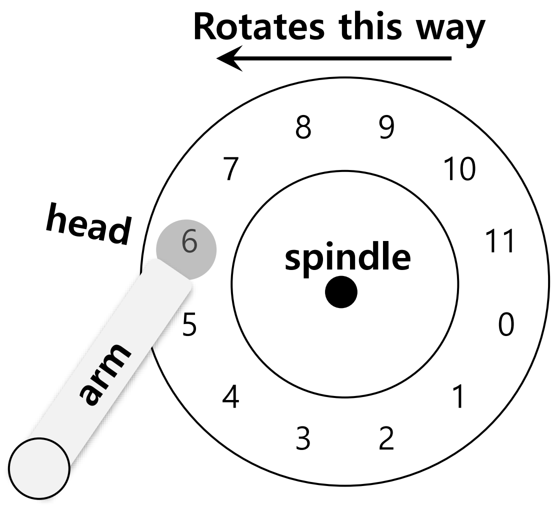
**A Disk with Just A Single Track (12 sectors)**

- **Platter** (Aluminum coated with a thin magnetic layer)
  - A circular hard surface
  - Data is stored persistently by inducing magnetic changes to it.
  - Each platter has 2 sides, each of which is called a **surface**.

## Basic Geometry (Cont.)

- **Spindle**
  - Spindle is connected to a motor that spins the platters around.
  - The rate of rotations is measured in **RPM** (Rotations Per Minute).
    - Typical modern values : 7,200 RPM to 15,000 RPM.
    - E.g., 10000 RPM : A single rotation takes about 6 ms.
      - (10,000 rot/min)(1/60 min/second) = 166.67 rot/sec
      - 1/166.67 sec/rot= .006 sec/rot
- **Track**
  - Concentric circles of sectors
  - Data is encoded on each surface in a track.
  - A single surface contains thousands of tracks.

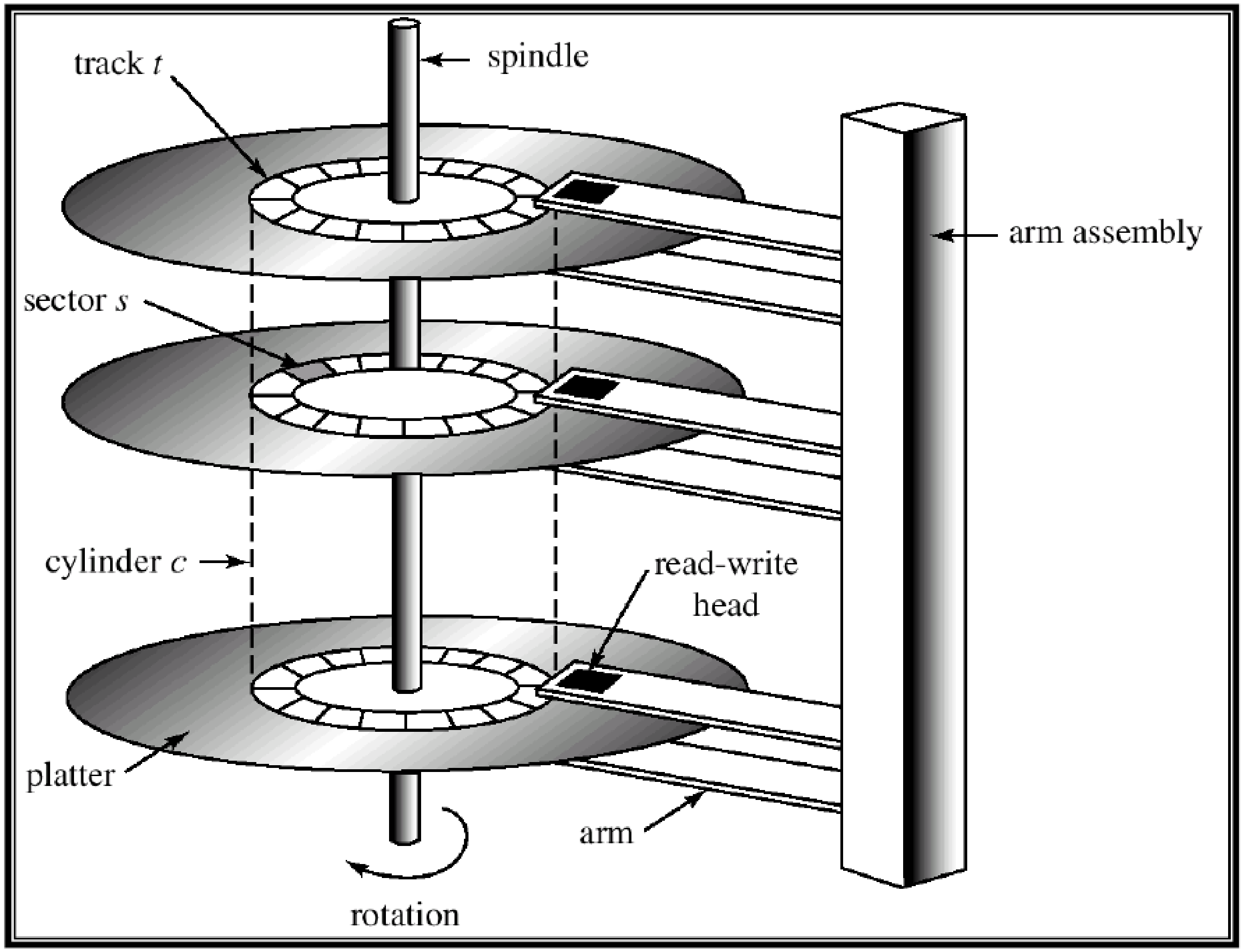
## A Simple Disk Drive



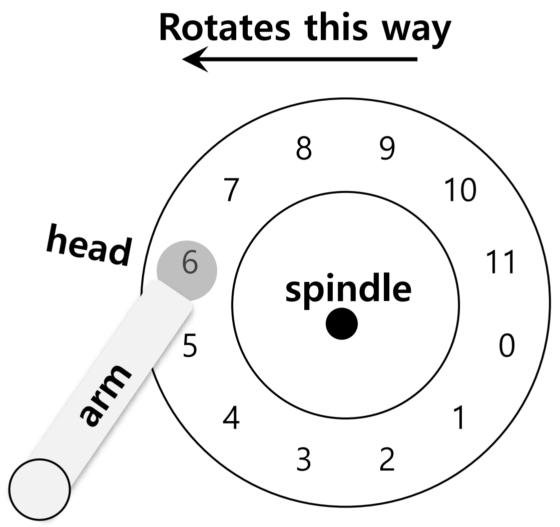
A Single Track Plus A Head

- **Disk head** (One head per surface of the drive)
  - The process of *reading* and *writing* is accomplished by the **disk head**.
  - Attached to a single disk arm, which moves across the surface.

## Example of a Disk



# Single-track Latency: The Rotational Delay



A Single Track Plus A Head

- **Rotational delay:** Time for desired sector to rotate under the head
  - Ex) Full rotational delay is `R` and we start at `sector 6`
    - Read sector 0: Rotational delay = $\frac{R}{2}$
    - Read sector 5: Rotational delay = `R-1` (worst case.)

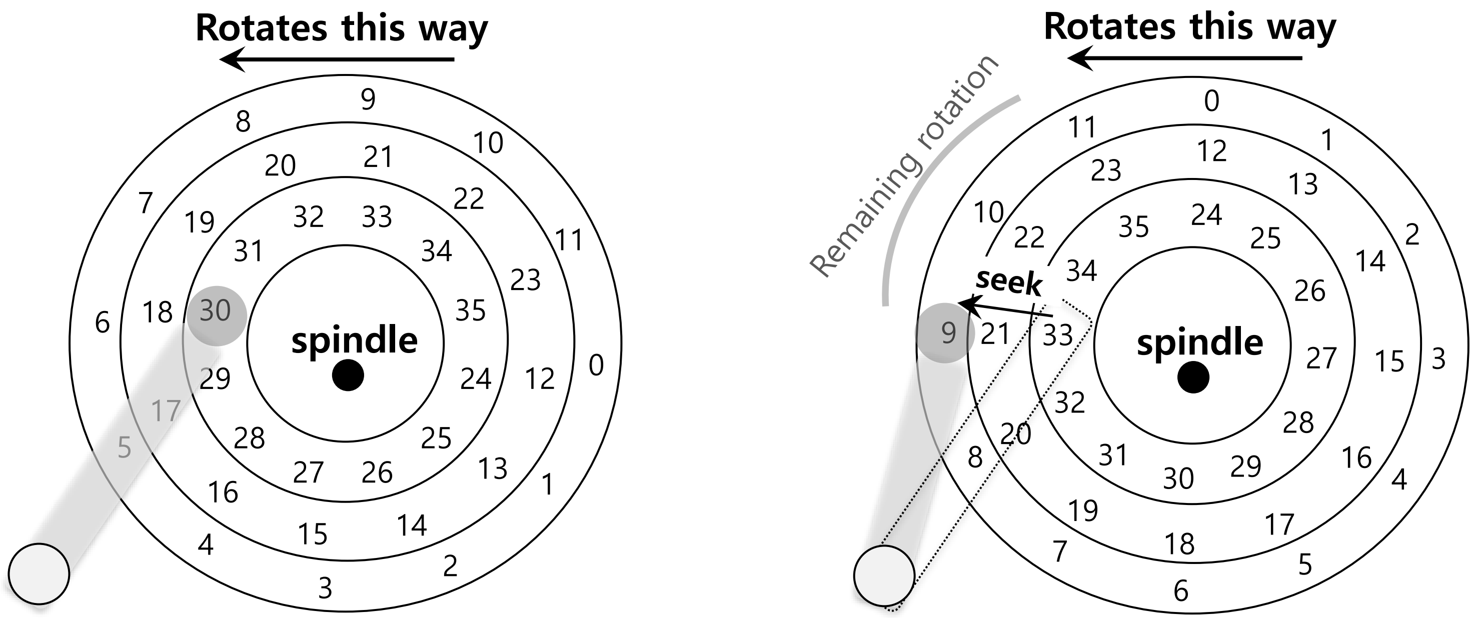
## Transfer

- The final phase of I/O
  - Data is either *read from* or *written* to the surface.

- Complete I/O time:
  - **Seek**
  - Waiting for the **rotational delay**
  - **Transfer**

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

# Multiple Tracks: Seek Time



**Three Tracks Plus A Head (Right: With Seek)**
**(e.g., read to sector 11)**

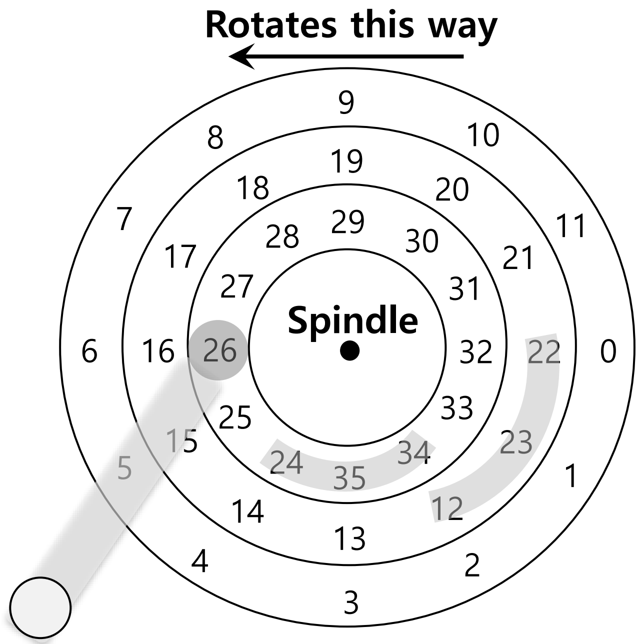
- **Seek**: Move the disk arm to the correct track
  - **Seek time**: Time to move head to the track containing the desired sector.
  - One of the most costly disk operations.

## Phases of Seek

- Why is it costly?
- Acceleration → Coasting → Deceleration → Settling
  - **Acceleration**: The disk arm gets moving.
  - **Coasting**: The arm is moving at full speed.
  - **Deceleration**: The arm slows down.
  - **Settling**: The head is *carefully positioned* over the correct track.
    - The settling time is often quite significant, e.g., 0.5 to 2ms.

## Track Skew

- Make sure that sequential reads can be properly serviced **even when crossing track boundaries**.



When reading sequential sectors see how 11 and 12 are offset? Same with 23 and 24.

Three Tracks: Track Skew Of 2

- *Without track skew*, the head would be moved to the next track but the desired next block would have already rotated under the head.

## Cache (Track Buffer)

- **Holds data** read from or written to the disk
  - Allow the drive to quickly respond to requests.
  - Small amount of memory (usually around 8 or 16 MB)

## Write on cache

- **Writeback** (Immediate reporting)
  - Acknowledge a write has completed when it has **put the data in the drives  memory (not yet on the disk but cached and ready to be written)** .
  - faster but dangerous (what if disk problem?)

- **Write through**
  - Acknowledge a write has completed after the write has **actually been written to disk**.

## I/O Time: Doing The Math

- I/O time ($T_{I/O}$): $T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$

- The rate of I/O ($R_{I/O}$): $R_{I/O} = \frac{Size_{Transfer}}{T_{I/O}}$

| | 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 |

**Disk Drive Specs: SCSI Versus SATA**

## I/O Time Example

- **Random workload**: Issue 4KB read to random locations on the disk
- **Sequential workload**: Read 100MB consecutively from the disk

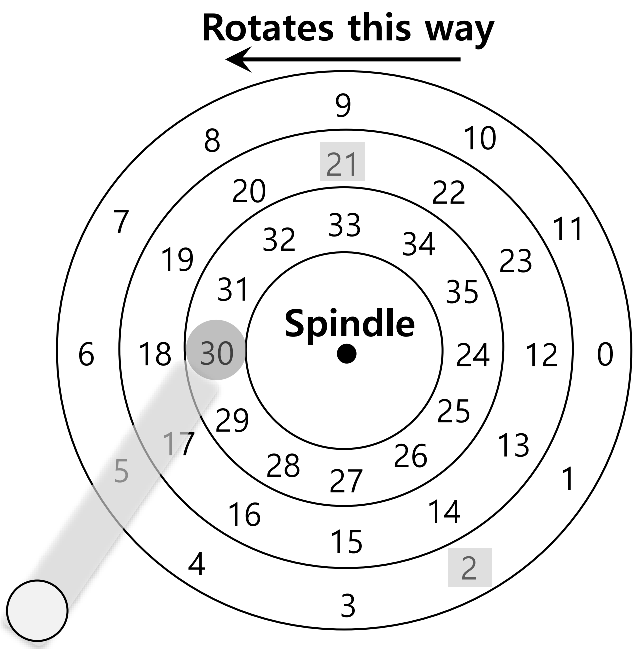
| | | **Cheetah 15K.5** | **Barracuda** |
|---|---|---|---|
| $T_{seek}$ | | 4 ms | 9 ms |
| $T_{rotation}$ | | 2 ms | 4.2 ms |
| Random | $T_{transfer}$ | 30 microsecs | 38 microsecs |
| | $T_{I/O}$ | 6 ms | 13.2 ms |
| | $R_{I/O}$ | 0.66 MB/s | 0.31 MB/s |
| Sequential | $T_{transfer}$ | 800 ms | 950 ms |
| | $T_{I/O}$ | 806 ms | 963.2 ms |
| | $R_{I/O}$ | 125 MB/s | 105 MB/s |

**Disk Drive Performance: SCSI Versus SATA**

**There is a huge gap in drive performance between random and sequential workloads**

## Disk Scheduling

- **Disk Scheduler** decides <u>which I/O request</u> to schedule next.
- **SSTF** (Shortest Seek Time First)
  - Order the queue of I/O request by track
  - Pick requests on the nearest track to complete first



Unlike job scheduling, we know roughly how long each disk request will take (estimate the seek and rotational delay then use xfer rate and number bytes to xfer)

**SSTF: Scheduling Request 21 and 2**

**Issue the request to 21 → issue the request to 2**

## SSTF problems.

- **Problem 1**: The drive geometry is not available to the host OS (OS isn't aware of sector layout on tracks, does not know which tracks are near to each other)
  - Solution: OS can simply implement <u>Nearest-block-first</u> (NBF)

  If need to access sector 1, 9, 512 and 15 then do 1,9,15,512

- **Problem 2**: Starvation
  - If there were a steady stream of request to the inner track, request to other tracks would then be ignored completely.

## I/O merging

- **Reduce the number of request** sent to the disk and lowers overhead
  - E.g., read blocks 33, then 8, then 34:
    - The scheduler merge the request for blocks 33 and 34 *into a single two-block request*.