KYUNGPOOK NATIONAL UNIVERSITY

Hard Disk Drives

persistence.

I/O Device 중 하나 HDD

Prof. Yongtae Kim

Computer Science and Engineering
Kyungpook National University

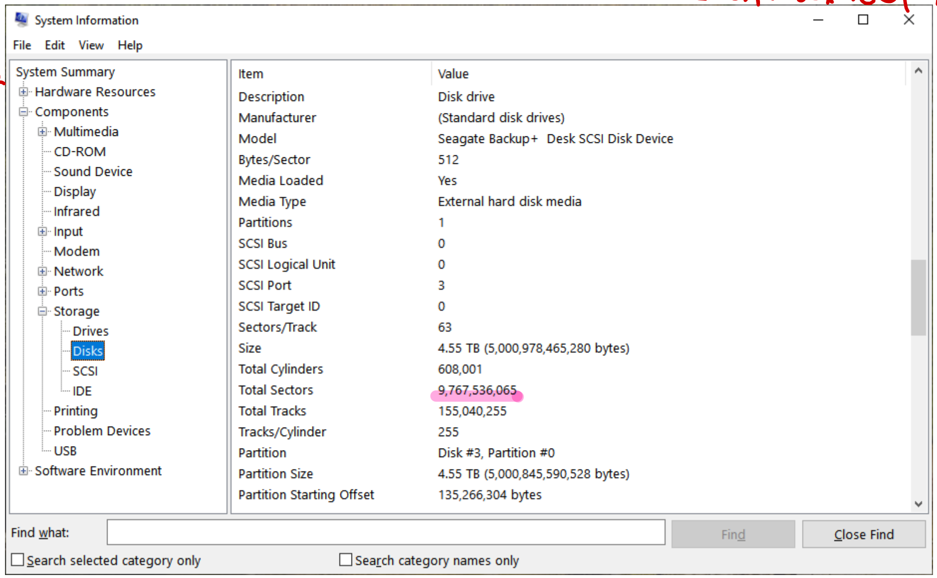
The Interface

HDD.

- **Hard disk drives have been the main form of persistent data storage in computer systems for decades**
  - The drive consists of huge number of sectors (512-byte blocks), each of which can be read or written, and are numbered from 0 to n − 1 on a disk (n sectors) n개
  - The disk is an array of sectors; 0 to n − 1 is thus the address space of the drive
  - Multi-sector operations: many file systems will read/write 4KB page at a time (or more)
  - A single 512-byte write is atomic; therefore, if an untimely power loss occurs, only a portion of a larger write may complete (sometimes called a torn write)

block들 묶어. 중간에 정전시 차단됨

512 바이트에 대한 읽고 쓰는 작업 원자적으로 발생



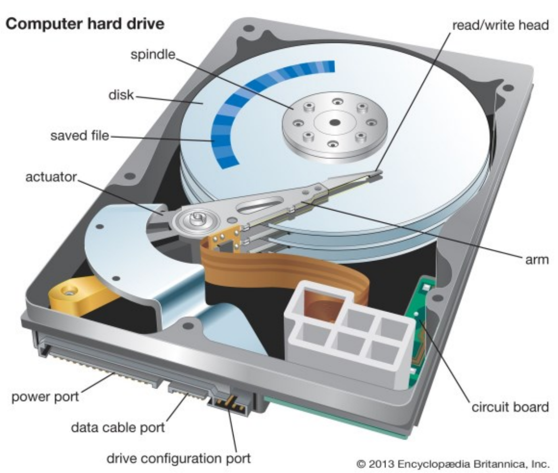
Basic Geometry

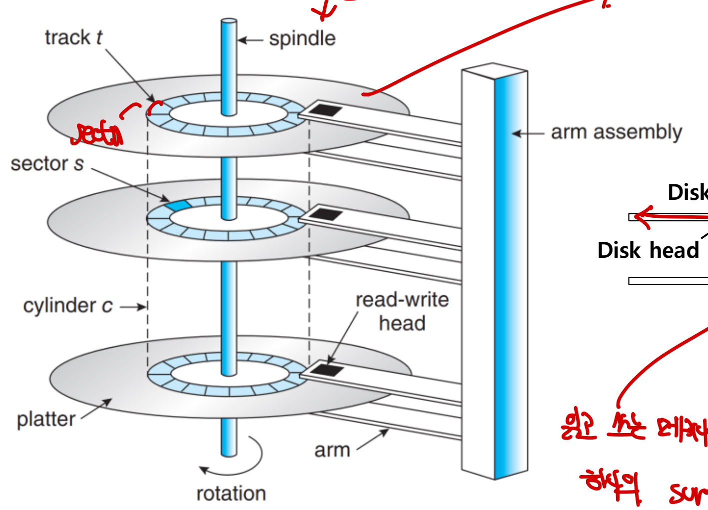
- **A modern hard disk drive includes some components**
  - A platter is a circular hard surface on which data is stored persistently by inducing magnetic changes to it and has two surfaces (2 or more platters)
  - A spindle is connected to a motor that spins the platters around at a constant (fixed) rate, often measured in rotations per minute (RPM) (7200~15000 RPMs)
  - A track is a concentric circle on surface, which contains many thousands tracks
  - Reading and writing is achieved by the disk head (one head per surface)
  - The disk head is attached to a single disk arm, which moves across the surface to position the head over the desired track

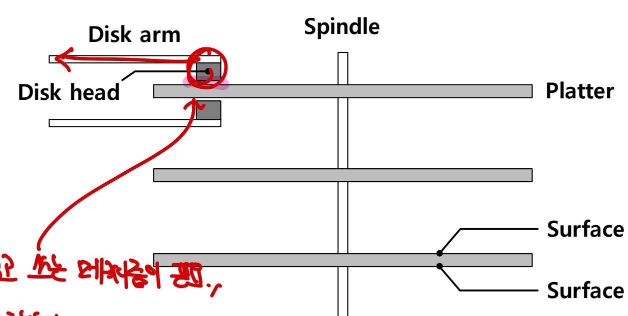
개 빨리 돔

플래터의 surface → 섹터들이 동심원으로 저장
= track.

즉 track은 여러개의 sector로







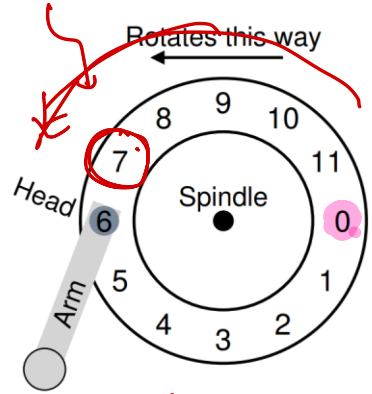
읽고 쓰는 메커니즘이 필요.,
하나의 surface 당 하나의 헤드

A Simple Disk Drive

원판은 돌고 head가 위치할 때 걸리는 시간.

512B

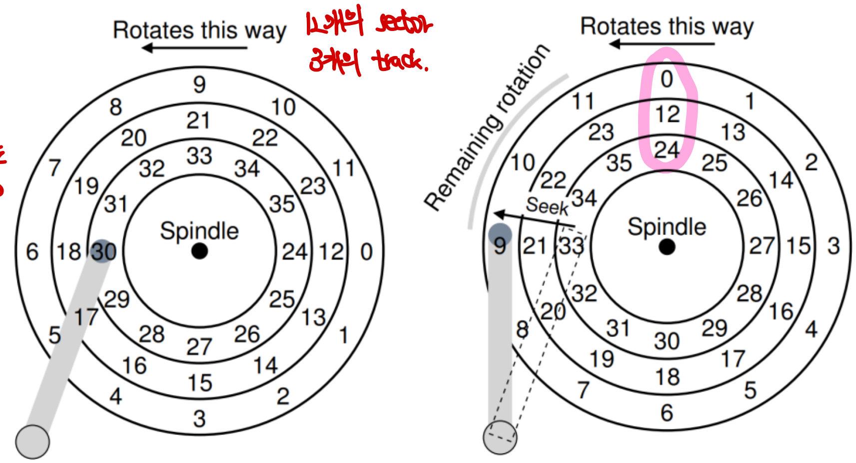
- **Assume a disk with a track of 12 sectors (0~11)**
  - To read, wait for the desired sector to rotate under disk head
  - This wait is an important I/O service time: rotational delay → 도는 데 걸리는 시간
  - If the full rotational delay is R, the disk has to incur a delay of about R/2 to wait for 0 to come under the head if we start at 6



head가 도는게 아니라 spindle는 움직임

- **Consider a disk with three tracks, each has 12 sectors**
  - seek + rotate 중요!
  - To read, the drive has to first move the disk arm to the correct track: seek (track에 갖다 놓음)
  - Multiple seek phases: acceleration → coasting → deacceleration → settling
  - The settling time is significant: 0.5~2ms
  - The platter is rotating during the seek → shorter rotational delay
  - The final I/O is the transfer, where data is read from or written to the surface → head를 통해 데이터를 읽음
  - Disk I/O: seek + rotate + transfer

12개의 sector 3개의 track.

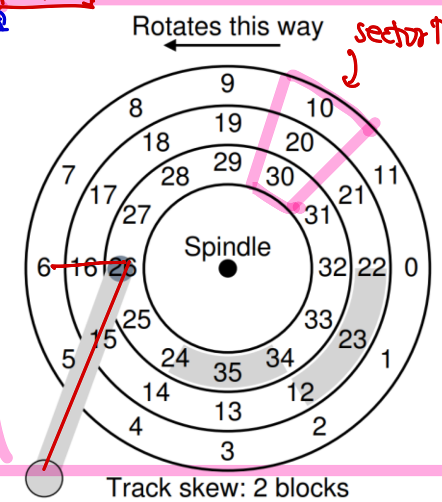


Some Other Details

OS write buffer 8~16MB
write 됐다 알림

연속적인 읽기가 잘될 수 있도록

- **Many drives employ some kind of track skew**
  - Making sure that sequential reads can be properly serviced even when crossing track boundaries
    track이 바뀔때 잘 읽힐 수 있도록
  - Without such skew, the head would be moved to the next track but the desired next block would have already rotated under the head



Track skew: 2 blocks

- **Outer tracks tend to have more sectors than inner tracks**
  - These tracks are often referred to as multi-zoned disk drives, where the disk is organized into multiple zones; outer zones have more sectors than inner zones

- **Important part of modern disk is its cache, called track buffer**
  - This cache is some small amount of memory (8 or 16 MB) which the drive can use to hold data read from or written to the disk
  - The write back caching acknowledges the write has completed when it has put the data in its memory buffer (faster but dangerous) → 실제 disk에는 쓰여지지 않을 수 있음
  - The write through caching acknowledges after the write has been written to disk
    → 가속도↓

I/O Time: Doing the Math

- **I/O time can be represented by** $T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$
  - I/O rate $R_{I/O}$ is computed from the time and it uses for comparison between drives $R_{I/O} = \frac{Size_{Transfer}}{T_{I/O}}$
- **Consider two workloads: random and sequential (4KB and 100MB read)**
  - We also consider two hard disk drives: high-end performance and low-end capacity

	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

Cheeta. Barracuda

  - On Cheetah with 4KB read:
    T_{seek} = 4ms, $T_{rotation}$ = 15000RPM=250RPS → 1/250=0.004s=4ms per rotation → 2ms on average (half rotation), $T_{transfer}$ = 4KB/125MB=0.00003125≈30us
    ➔$T_{I/O}$ = 4ms+2ms+30us ≈ 6ms, $R_{I/O}$=4KB/6ms ≈ 0.66MB/s
  - On Barracuda with 4KB read:
    T_{seek} = 9ms, $T_{rotation}$ = 7200RPM=120RPS→1/120≈8.3ms per rotation → 4.15ms on average, $T_{transfer}$ = 4KB/105MB ≈38us ➔$T_{I/O}$ ≈ 13.2ms, $R_{I/O}$ ≈ 0.31MB/s
  - With 100MB sequential read:
    $T_{transfer,cheetah}$ = 100MB/125MB=800ms,
    $T_{transfer,barracuda}$ = 100MB/105MB=950ms

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

Disk Scheduling

I/O는 비용↑, OS는 디스크에 발생하는 I/O 순서들을 효율적인 순서로 결정하는 역할을 함
=디스크 스케줄러
FCFS (First-come, first-serve)

- **Due to high cost of I/O, given I/O requests, the disk scheduler examines the requests and decides which one to schedule next**
  - Unlike job scheduling, where the length of each job is usually unknown, with disk scheduling, we can make a good guess at how long a disk request will take

=arm를 움직이는 데 걸리는 시간

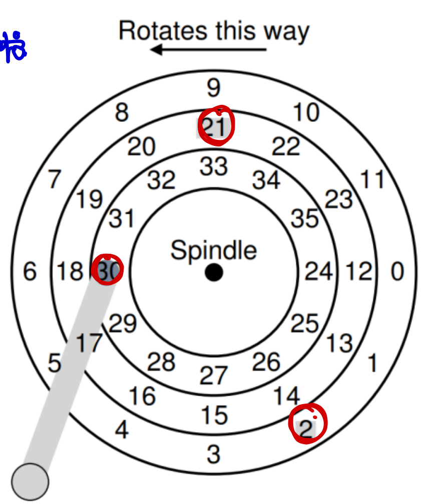
- **One early disk scheduling is shortest-seek-time-first (SSTF)**
  - SSTF orders the queue of I/O requests by track, picking requests on the nearest track to complete first

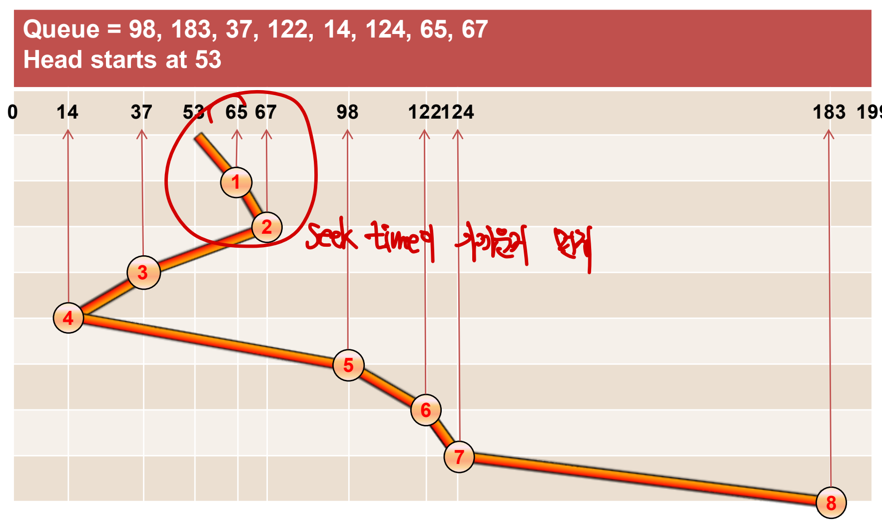
프로세스 스케줄링은 언제 끝날지를 모름.
but HDD는 예측 할 수 있음

  - SSTF is not a panacea due to 1) OS does not know the drive geometry, (thus, OS can implement nearest-block-first; NBF by nearest-address) 2) starvation

SSTF Scheduling:
30(current head)
→ 21 → 2

10 → 2? 21? 더 가까움

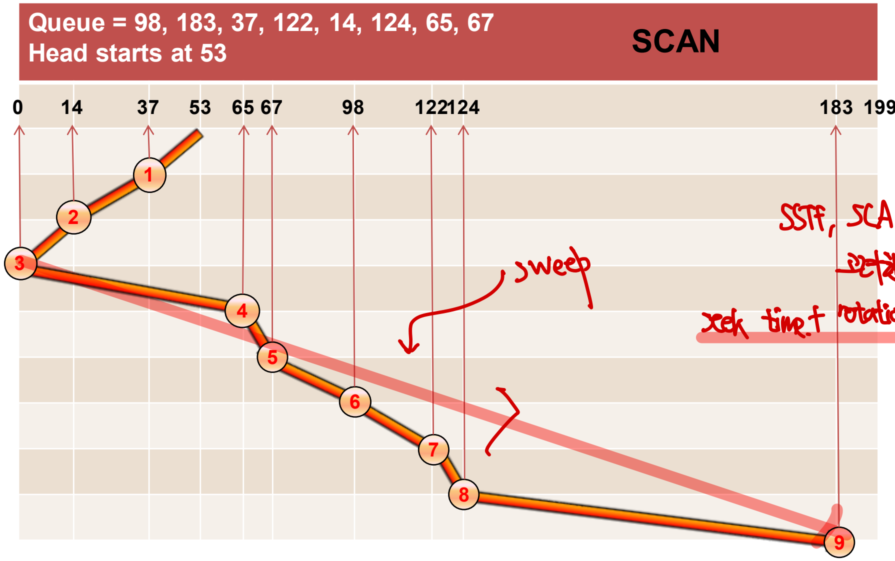




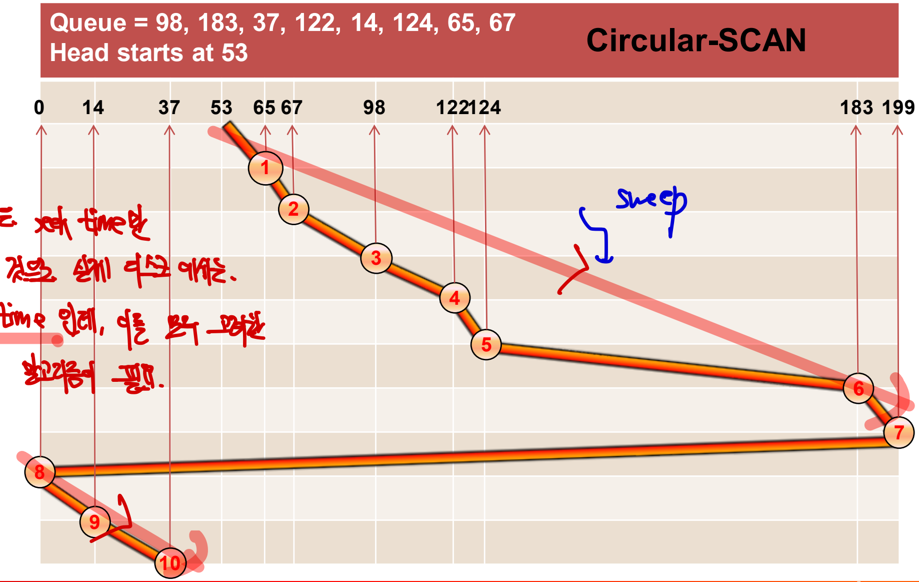
Elevator (a.k.a. SCAN or C-SCAN)

→starvation 해결.

- **The SCAN algorithm simply moves back and forth across the disk servicing requests in order across the tracks**
  - 1층 ⇄ 10층
  - Sweep: a single pass across the disk from outer/inner to inner/outer
  - if a request comes on a track that has already been serviced on this sweep, it is not handled immediately, but queued until the next sweep (in the other direction)
  - →새로운 데이터들 큐에 추가 x →기아 문제를 해결할 수 있음.
  - F-SCAN freezes the queue to be serviced when it is doing a sweep
  - circular / 한 방향으로만
  - C-SCAN only sweeps from outer-to-inner, and resets at the outer to begin again
  - SCAN is sometimes referred to as the elevator algorithm
  - 안쪽과 바깥쪽 트랙에 대해 좀더 공평하게 처리할 수 있음



SSTF, SCAN은 seek time만 고려한 것으로 실제 디스크 에서는 seek time+rotation time 인데, 이를 모두 고려한 알고리즘이 필요.



Shortest Positioning Time First & Other Issues

arm의 위치. seek time, rotation time을 고려하자.

- **Shortest Positioning Time First (SPTF) takes into account the head position, particularly, seek time and rotational delay**
  - e.g.) The head is positioned over 30 on the inner track and the scheduler has to decide which should it service next? Sector 16 (middle track) or 8 (outer track)

    SSTF는 30→16→8
  - The answer is, "it depends" → consider both seek time and rotational delay
  - 16 → 8: 1 seek + 10/12 rotation + 1 seek + 4/12 rotation (2 seek + 7/6 rotation)
  - 8 → 16: 1 seek↑ + 2/12 rotation + 1 seek + 8/12 rotation (2 seek↑ + 5/6 rotation)

    2 track 변경 HDD 내부 작동
  - SPTF selects a request which has the smallest position time (seek + rotational delay) → It is difficult to implemented in OS and thus performed inside a drive

    OS가 하기 힘듦. arm의 위치가 어딘지 알기 힘듦
- **Other Scheduling Issues**
  - Where is disk scheduling performed? OS or disk

    HDD Driver.
  - I/O merging? requests of 33, 8, 34 → merging 33, 34

    묶어서.
  - How long should the system wait before issuing an I/O to disk? Immediately (work-conserving) or waiting for a bit (non-work-conserving) → waiting is better due to a new better request may arrives at the disk

    추후 기다림에 가능. (merging 할 가능성↑)

    =실제로 더 빠름