Lecture 11
Storage

Prof. Yinqian Zhang

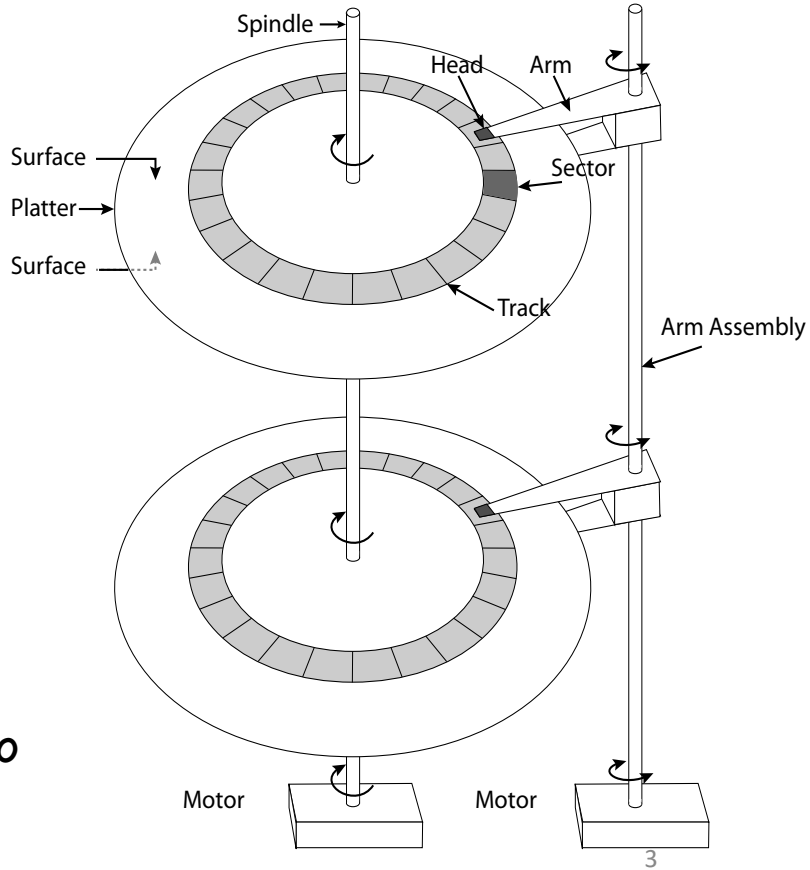
Spring 2023

Storage Devices

- Magnetic disks
  - Storage that rarely becomes corrupted
  - Large capacity at low cost
  - Block level random access
  - Poor performance for random access
  - Better performance for sequential access
- Flash memory
  - Storage that rarely becomes corrupted
  - Capacity at intermediate cost (5-20x disk)
  - Block level random access
  - Good performance for reads; worse for random writes

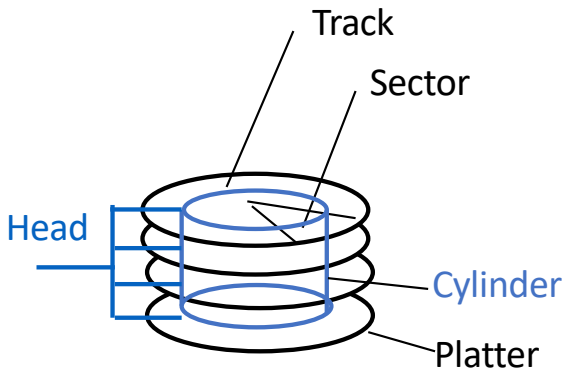
The Amazing Magnetic Disk

- Unit of Transfer: Sector
  - Ring of sectors form a track
  - Stack of tracks form a cylinder
  - Heads position on cylinders
- Disk Tracks ~ 1μm (micron) wide
  - Wavelength of light is ~ 0.5μm
  - Resolution of human eye: 50μm
  - 100K tracks on a typical 2.5" disk
- Separated by unused guard regions
  - Reduces likelihood neighboring tracks are corrupted during writes (still a small non-zero chance)

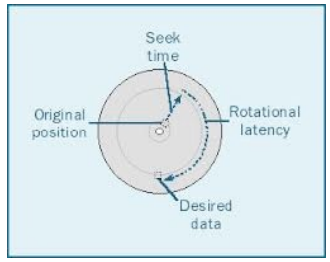


Magnetic Disks

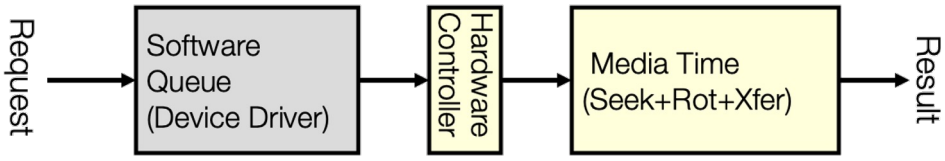
- Cylinders: all the tracks under the head at a given point on all surface
- Read/write data is a three-stage process:
  - Seek time: position the head/arm over the proper track
  - Rotational latency: wait for desired sector to rotate under r/w head
  - Transfer time: transfer a block of bits (sector) under r/w head



Seek time = 4-8ms
One rotation = 1-2ms
(3600-7200 RPM)



Disk Latency = Queuing Time + Controller time + Seek Time + Rotation Time + Transfer Time



Disk Performance Example

- Assumptions:
  - Ignoring queuing and controller times for now
  - Avg seek time of 5ms,
  - 7200RPM $\Rightarrow$ Time for rotation: 60000 (ms/minute) / 7200(rev/min) $\cong$ 8ms
  - Transfer rate 4MByte/s, sector size 1 Kbyte $\Rightarrow$ 1024 bytes/4×10^6 (bytes/s) = 256×10^{-6} sec $\cong$ .26 ms
- Read sector from random place on disk:
  - Seek (5ms) + Rot. Delay (4ms) + Transfer (0.26ms)
  - *Approx* 10ms to fetch/put data: 100 KByte/sec
- Read sector from random place in the same cylinder:
  - Rot. Delay (4ms) + Transfer (0.26ms)
  - *Approx* 5ms to fetch/put data: 200 KByte/sec
- Read next sector on same track:
  - Transfer (0.26ms): 4 MByte/sec

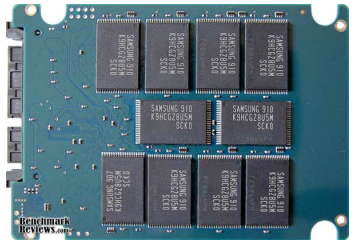
Key to using disk effectively (especially for file systems) is to *minimize seek and rotational delays*

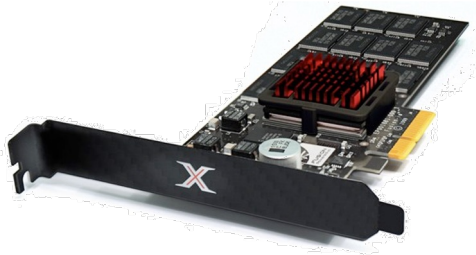
Typical Numbers for Magnetic Disk

Parameter	Info / Range
Space/Density	Space: 8TB (Seagate), 10TB (Hitachi) in 3½ inch form factor! Areal Density: ≥ 1Terabit/square inch! (SMR, Helium, …)
Average seek time	Typically 5-10 milliseconds. Depending on reference locality, actual cost may be 25-33% of this number.
Average rotational latency	Most laptop/desktop disks rotate at 3600-7200 RPM (16-8 ms/rotation). Server disks up to 15,000 RPM. Average latency is halfway around disk so 8-4 milliseconds
Controller time	Depends on controller hardware
Transfer rate	Typically 50 to 100 MB/s.
Cost	Used to drop by a factor of two every 1.5 years (or even faster); now slowing down

Solid State Disks (SSDs)

- 1995 – Replace rotating magnetic media with non-volatile memory
- 2009 – Use NAND Multi-Level Cell (2 or 3-bit/cell) flash memory
  - Sector (4 KB page) addressable, but stores 4-64 "pages" per memory block
  - Trapped electrons distinguish between 1 and 0
  - Data erased at the block level
- No moving parts (no rotate/seek motors)
  - Eliminates seek and rotational delay (0.1-0.2ms access time)
  - Very low power and lightweight
  - Limited "write cycles"
- Rapid advances in capacity and cost ever since!



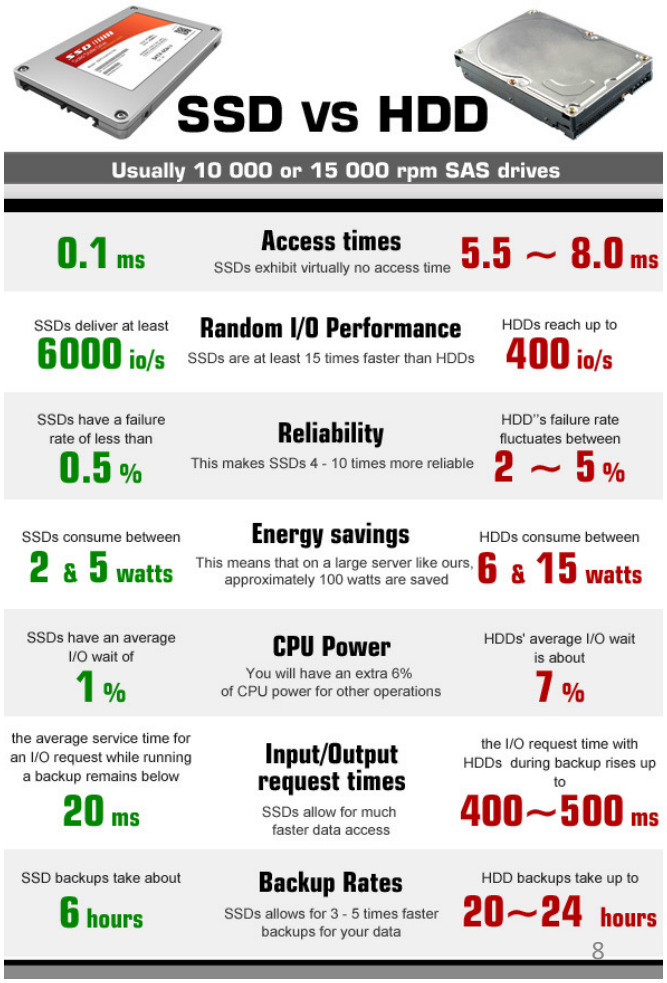


HDD vs SSD Comparison

SSD prices drop much faster than HDD

Price Crossover Point for HDD and SSD

	2012	2013	2014	2015E	2016F	2017F
HDD	0.09	0.08	0.07	0.06	0.06	0.06
2.5" SSD	0.99	0.68	0.55	0.39	0.24	0.17

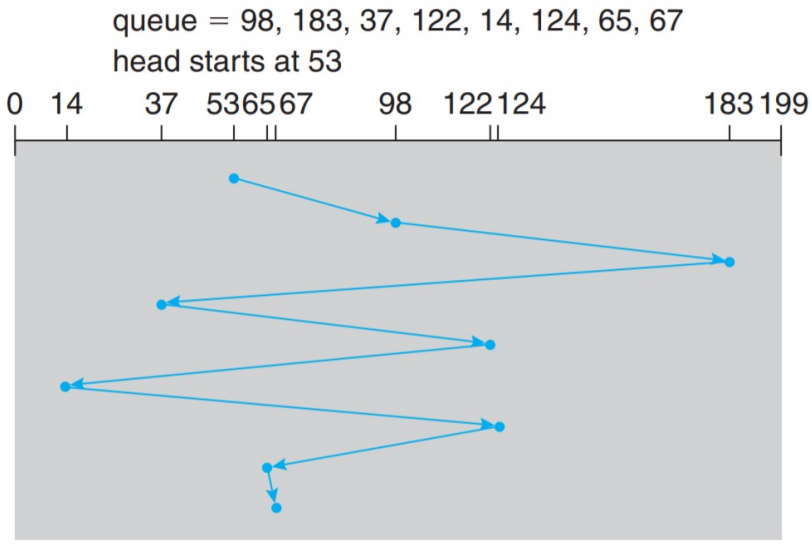


Disk Scheduling

- There are many sources of disk I/O request
  - OS
  - System processes
  - User processes
- OS should think how to use hardware efficiently?
  - Disk bandwidth
  - Access time
- Given a sequence of access cylinders in the HDD
  - 98, 183, 37, 122, 14, 124, 65, 67
  - Head point: 53
  - Pages: 0 ~ 199
- Minimize seek time
  - Seek time $\approx$ seek distance
- How to minimize the total head movement distance?
  - Minimize the total number of cylinders.

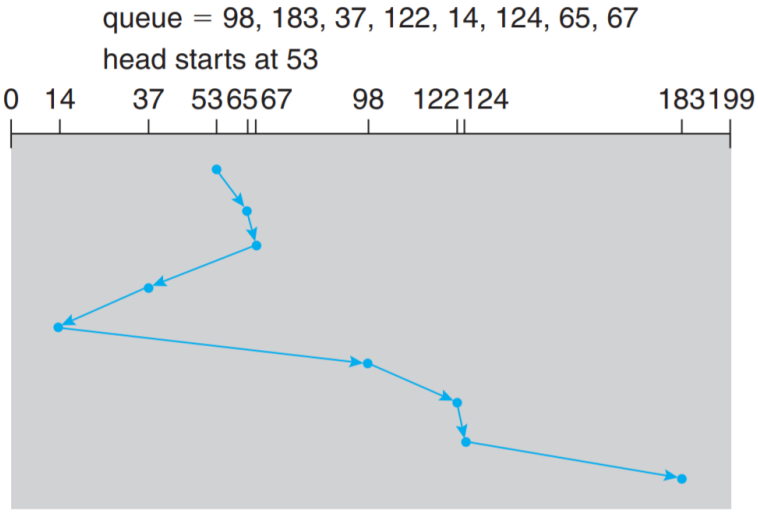
Disk Scheduling: FIFO

- FIFO Order
  - Fair among requesters, but order of arrival may be to random spots on the disk ⇒ Very long seeks
- The head movement distance = ?



Disk Scheduling: SSTF

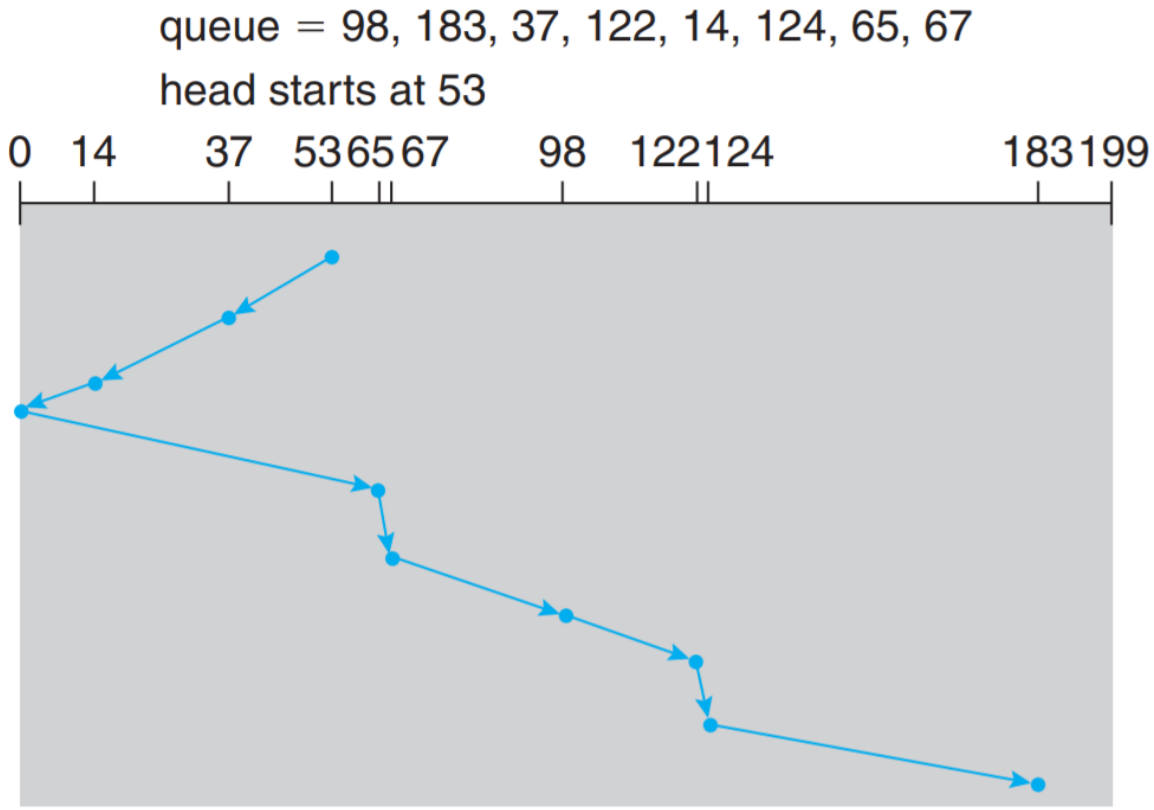
- Shortest Seek Time First order
  - Shortest Seek Time First selects the request with the minimum seek time from the current head position
  - SSTF scheduling is a form of **SJF** scheduling; may cause **starvation** of some requests
- The head movement distance = ?



Disk Scheduling: SCAN

- SCAN order
  - SCAN algorithm a.k.a., elevator algorithm
  - The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
  - But note that if requests are uniformly dense, largest density at other end of disk and those wait the longest
- The head movement distance = ?

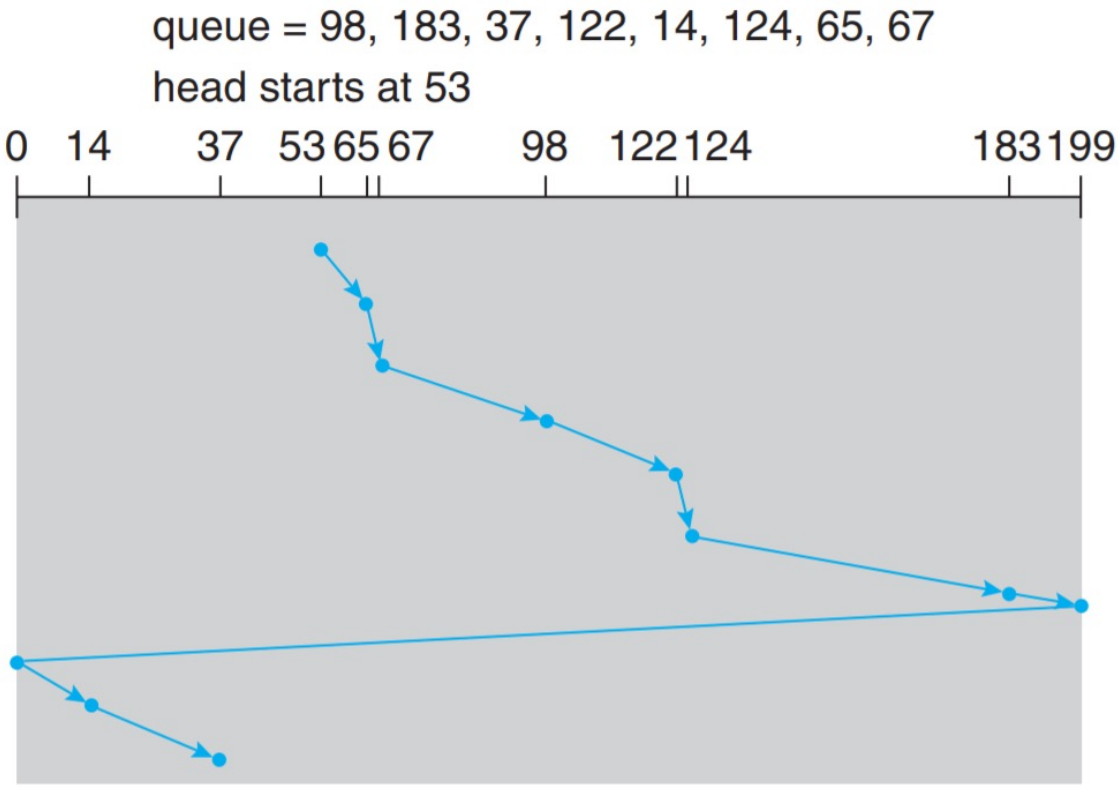
Disk Scheduling: SCAN



Disk Scheduling: C-SCAN

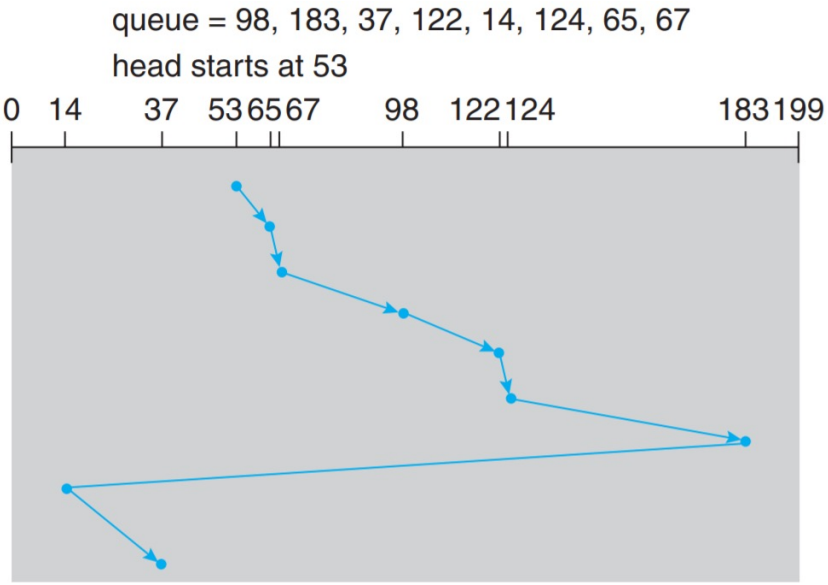
- C-SCAN order
  - Provides a more uniform wait time than SCAN
  - The head moves from one end of the disk to the other, servicing requests as it goes
    - When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip
  - Treats the cylinders as a circular list that wraps around from the last cylinder to the first one
- The head movement distance =?

Disk Scheduling: C-SCAN



Disk Scheduling: LOOK, C-LOOK

- Look and C-LOOK order
  - LOOK a variant of SCAN, C-LOOK a variant of C-SCAN
  - Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk
- C-LOOK: the head movement distance = ?



Selecting a Disk-Scheduling Algorithm

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk
  - Less starvation
- Either SSTF or LOOK is a reasonable choice for the default algorithm
- Performance depends on the number and types of requests
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary

Thank you!