

Computing Infrastructure



POLITECNICO DI MILANO



Storage: Disk abstraction and HDD

POLITECNICO DI MILANO

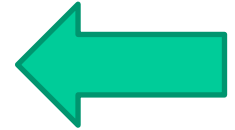


The topics of the course: what are we going to see today?



A. HW Infrastructures:

- **System-level**: Computing Infrastructures and Data Center Architectures, Rack/Structure;
- **Node-level**: Server (computation, HW accelerators), **Storage (Type, technology)**, Networking (architecture and technology);
- **Building-level**: Cooling systems, power supply, failure recovery



B. SW Infrastructures:

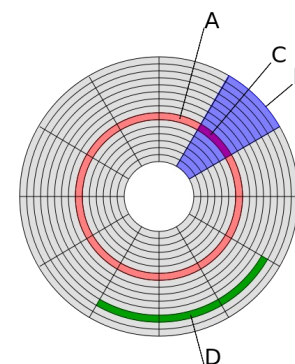
- **Virtualization**: Process/System VM, Virtualization Mechanisms (Hypervisor, Para/Full virtualization)
- **Computing Architectures**: Cloud Computing (types, characteristics), Edge/Fog Computing, X-as-a service
- **Machine and deep learning-as-a-service**

C. Methods:

- **Reliability and availability of datacenters** (definition, fundamental laws, RBDs)
- **Disk performance** (Type, Performance, RAID)
- **Scalability and performance of datacenters** (definitions, fundamental laws, queuing network theory)



- Disks can be seen by an OS as a collection of *data blocks* that can be read or written independently.
- To allow the ordering/management among them, each block is characterized by a unique numerical address called *LBA (Logical Block Address)*.
- Typically, the OS groups blocks into *clusters* to simplify the access to the disk. Clusters are the minimal unit that an OS can read from or write to a disk.
- Typical cluster sizes range from 1 disk sector (512 B) to 128 sectors (64 KB).



(A) track (B) geometrical sector
(C) track sector (D) cluster



Clusters contains:

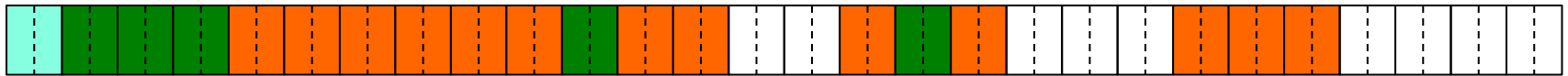
- **File data**: the actual content of the files
- **Meta data**: the information required to support the file system.

Meta data contains:

- File names
- Directory structures and symbolic links
- File size and file type
- Creation, modification, last access dates
- Security information (owners, access list, encryption)
- *Links to the LBA where the file content can be located on the disk*



The disk can thus contain several types of clusters:



Meta data – fixed position (to bootstrap the entire file system)



Meta data – variable position (to hold the folder structure)



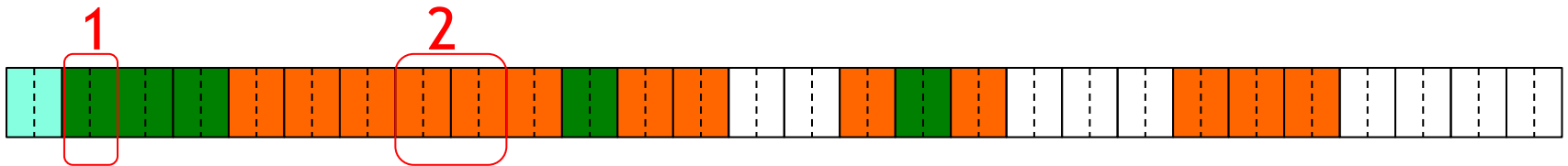
File data (actual content of the files)



Unused space (available to contain new files and folders)



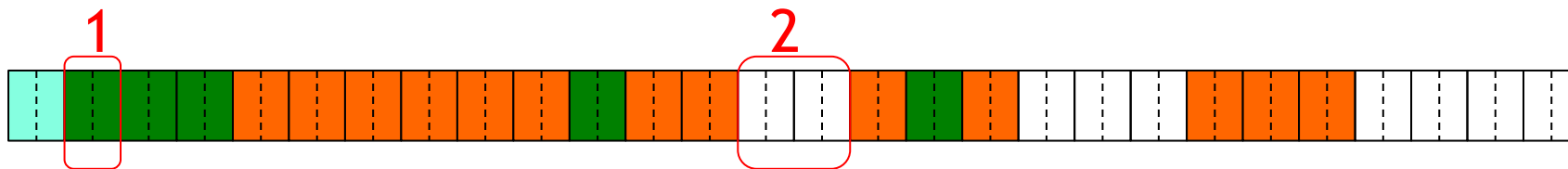
Reading a file requires to:



1. Accessing the meta-data to locate its blocks
2. Access the blocks to read its content



Writing a file requires to:



1. Accessing the meta-data to locate free space.
2. Write the data in the assigned blocks



Since the file system can only access clusters, the real occupation of space on a disk for a file is always a multiple of the cluster size.

Let us call:

- s – the file size
- c – the cluster size
- a – the actual size on disk

Then, we have:

$$a = \text{ceil}(s / c) * c$$

And the quantity $w = a - s$ is **wasted disk space** due to the organization of the file into clusters.

This waste of space is called *internal fragmentation* of files.



An example of internal fragmentation:

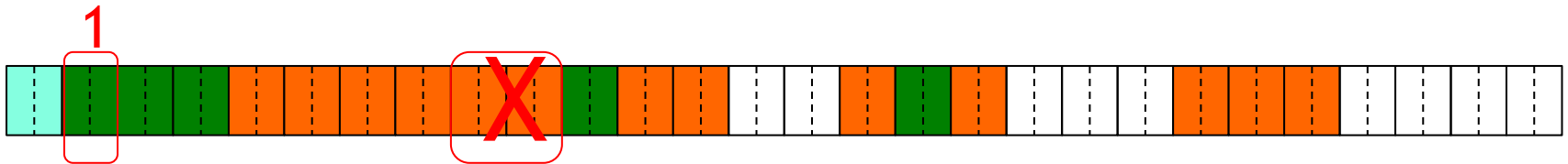
- **s - file size = 27 byte**
- **c – cluster size = 8 byte**
- **actual size on the disk**

$$a = \text{ceil}(27 / 8) * 8 = \text{ceil}(3.375) * 8 = 4 * 8 = \mathbf{32 \text{ byte}}$$

- ***Wasted disk space = 32 – 27 = 5 byte***



Deleting a file requires:



1. Only to update the meta-data to say that the blocks where the file was stored are no longer in use by the O.S.

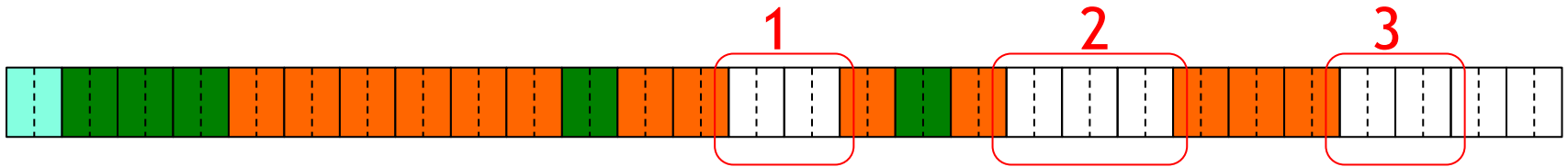
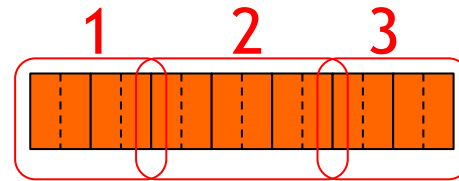
Deleting a file never actually deletes the data on the disk: when a new file will be written on the same clusters, the old data will be replaced by the new one.



Files - external fragmentation



As the life of the disk evolves, there might not be enough space to store a file contiguously.



In this case, the file is split into smaller chunks that are inserted into the free clusters spread over the disk.

The effect of splitting a file into non-contiguous clusters is called *external fragmentation*.

As we will see, this can reduce a lot the performance of an HDD.



HARD DRIVES



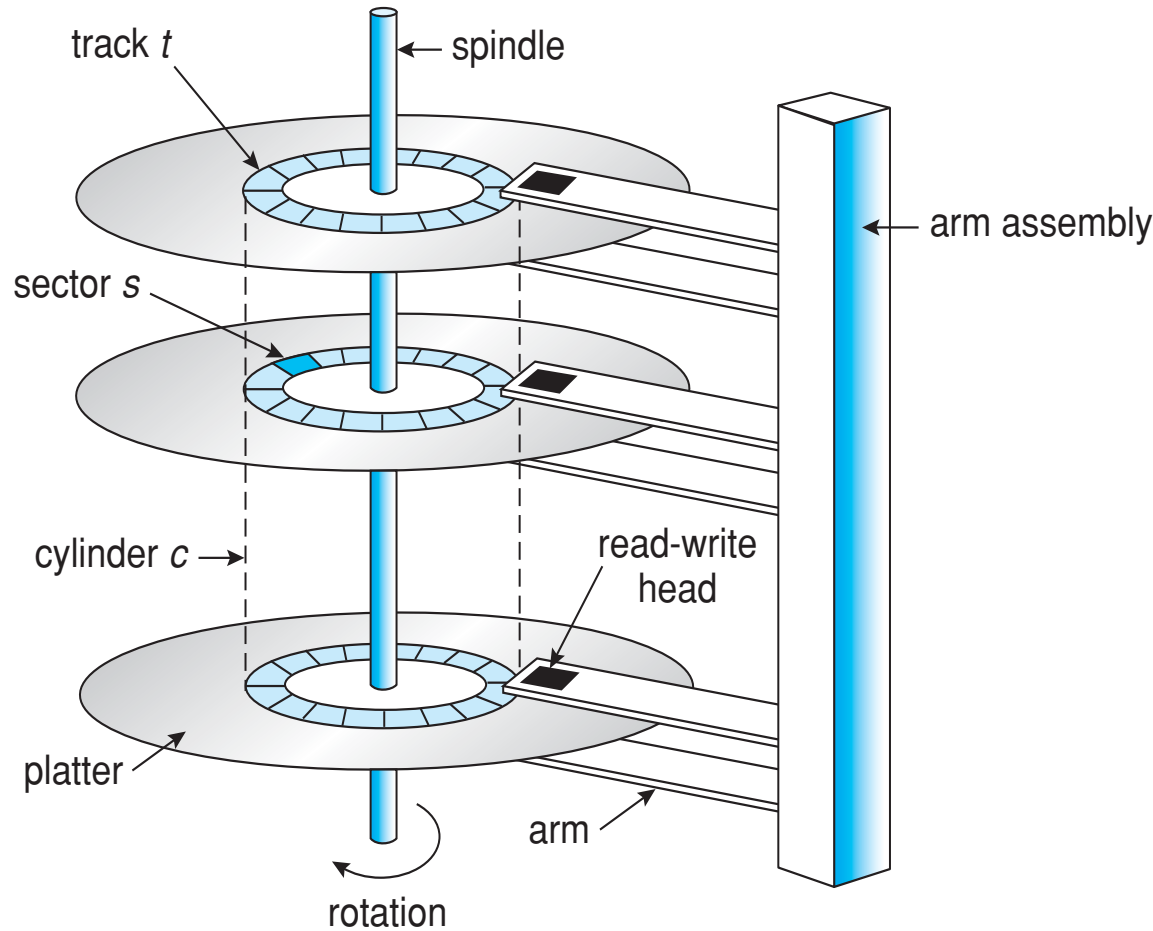
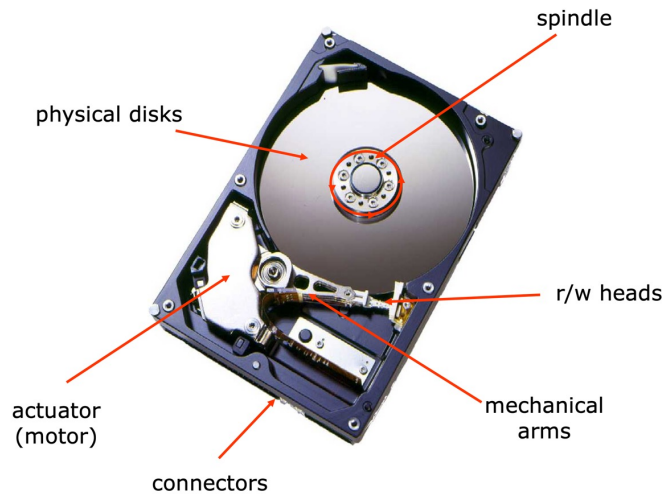
A hard disk drive (HDD) is a data storage using rotating disks (platters) coated with magnetic material.

Data is read in a random-access manner, meaning individual blocks of data can be stored or retrieved in any order rather than sequentially.

An HDD consists of one or more rigid ("hard") rotating disks (platters) with magnetic heads arranged on a moving actuator arm to read and write data to the surfaces.



Hard Drive anatomy: A Multi-Platter Disk





- Externally, hard drives expose a large number of **sectors** (blocks)
 - Typically 512 or 4096 bytes
 - Individual sector writes are **atomic**
 - Have a header and an error correction code
 - Multiple sectors writes may be interrupted (**torn write**)
 - *Torn writes* happens when only part of a multi-sector update are written successfully to disk
- Drive geometry
 - Sectors arranged into **tracks**
 - A **cylinder** is a particular track on multiple platters
 - Tracks arranged in concentric circles on **platters**
 - A disk may have multiple, double-sided platters
- Drive motor spins the platters at a constant rate
 - Measured in revolutions per minute (RPM)



- Diameter: about 9 cm (3,5 ÷ 2.5 in) - two surfaces
- Rotation speed: 7200 ÷ 15000 RPM round per minute
- Track density: 16,000 TPI (Track Per Inch)
- Heads: can be parked close to the center or to the outer diameter (mobile drives)
- Disk buffer cache: embedded memory in a hard disk drive that has the function of a buffer between the disk and the computer





Many disks incorporate caches (**track buffer**)

- Small amount of RAM (8, 16, or 32 MB)

Read caching

- Reduces read delays due to seeking and rotation

Write caching

- **Write back cache**: drive reports that writes are complete **after** they have been **cached**
 - Possibly dangerous feature. Inconsistent state if power goes off before the write back event
- **Write through cache**: drive reports that writes are **complete** after they have been **written** to **disk**

Today, some disks include flash memory for persistent caching (hybrid drives)



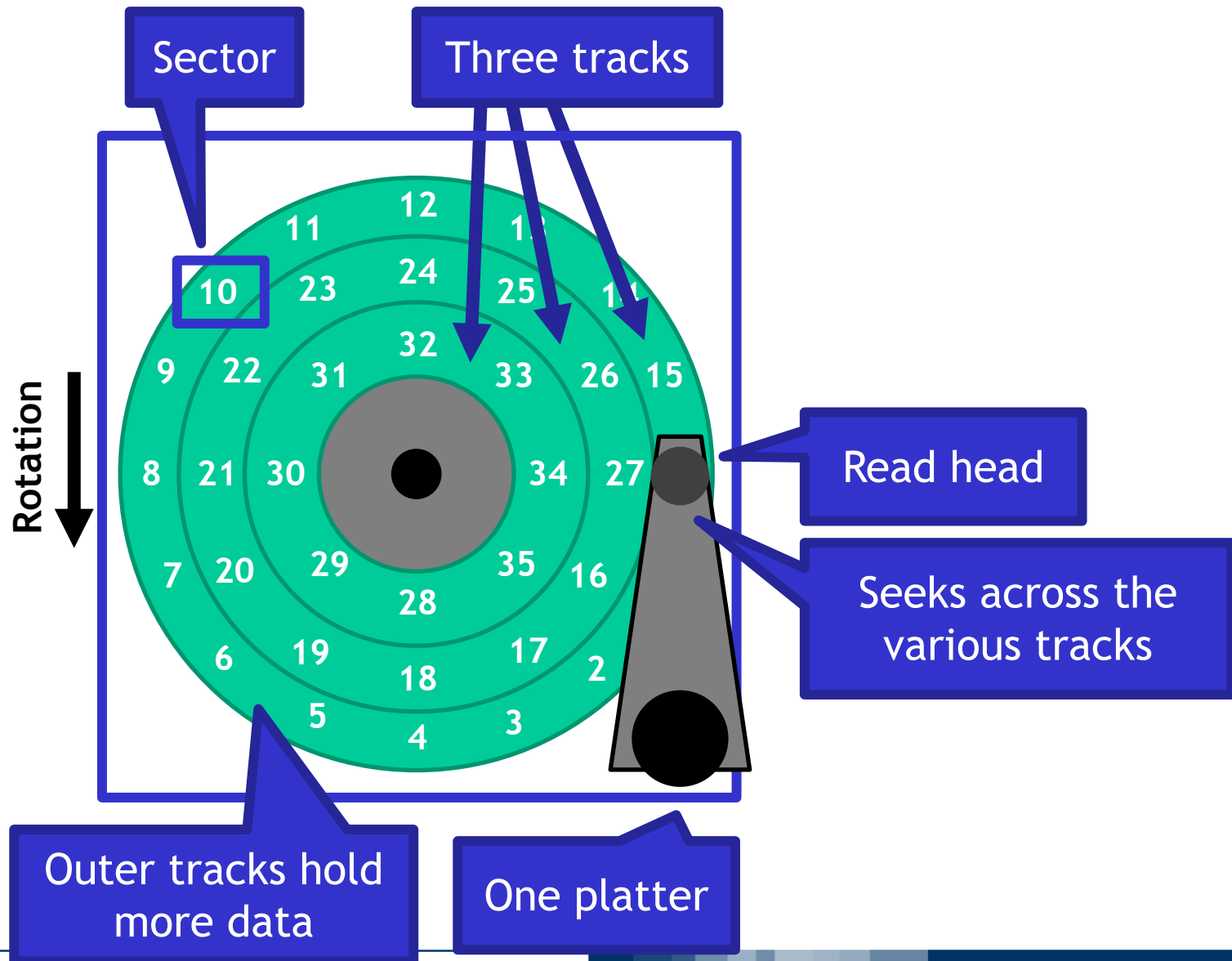
Some data concerning current Seagate HDDs

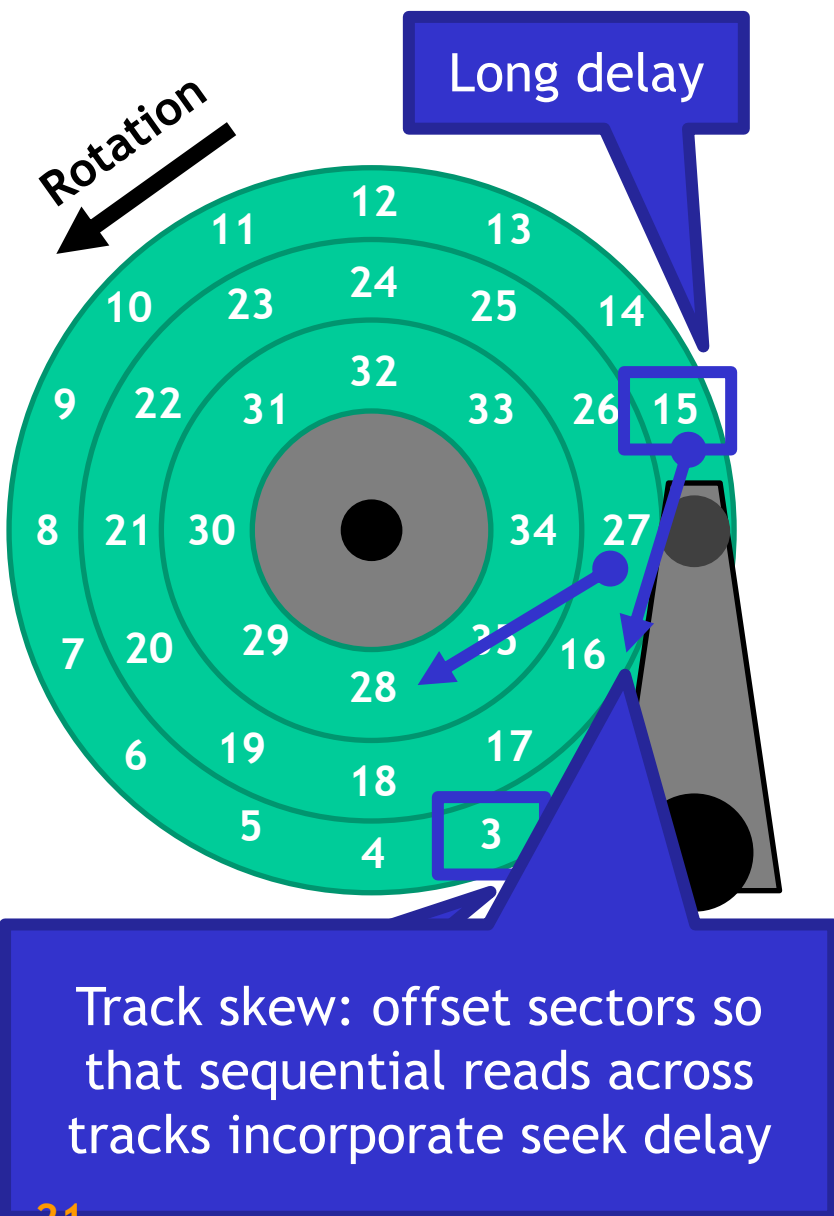


Specifications	Barracuda®	Constellation®	Constellation ES	SV35 Series™
Primary Applications	Optimised for PC and personal external storage	Optimised for 2.5-inch business server and external storage arrays	Optimised for 3.5-inch business server and external storage arrays	Optimised for video surveillance applications
Capacity (GB)	250, 320, 500, 750, 1,000 1,500, 2,000, 3,000	250, 500, 1,000	500, 1,000, 2,000, 3,000	1,000, 2,000, 3,000
Spin Speed (RPM)	7,200	7,200	7,200	7,200
SATA interface (Gb/s)	1.5/3.0/6.0	1.5/3.0/6.0	1.5/3.0/6.0	1.5/3.0/6.0
SAS interface (Gb/s)	—	3.0/6.0	3.0/6.0	—
Rotational vibration (RV) (radians/s/s)	5.5 narrow spectrum up to 300Hz	16 broad spectrum up to 1,800Hz	12.5 broad spectrum up to 1,500Hz	5.5 narrow spectrum up to 300Hz
Seek time, average read/write (ms)	<8.5/<9.5	8.5/9.5	8.5/9.5	<8.5/<9.5
Cache (MB) ¹	16, 64	Up to 64	Up to 64	64
Non-recoverable read errors per bits read	1 sector per 10 ¹⁴	1 sector per 10 ¹⁵	1 sector per 10 ¹⁵	1 sector per 10 ¹⁴
Power-on hours (POH)	2,400 – 8x5	8,760 - 24x7	8,760 - 24x7	8,760- 24x7
Streaming capabilities	—	Multiple sequential streams	Multiple sequential streams	Up to 20 simultaneous HD streams ³
POH usage profile	8x5 – On as needed	24x7 – Always on	24x7 – Always on	Up to 64 cameras 24x7 – Always on
MTBF (hours)	700,000	1.4 million	1.2 million	1 million
Power, average – idle (W) ²	4.6	2.25 to 3.85	>3.74	—
Power, average – Idle2 (W) ²	3.4 to 5.4	—	—	3.4 to 5.4
Acoustics, typical – idling (bels)	2.2 to 2.4	2.2	1.9 to 2.7	2.2 to 2.4
Shock, operating/non-operating (Gs)	70 to 80/300 to 350	70/400	40 to 70/300	80/300 to 350
Ambient temperature, operating/ non-operating (°C)	0 to 60/–40 to 70	5 to 60/–40 to 70	5 to 60/–40 to 70	0 to 70/–40 to 70
RAID support	0, 1	0, 1, 3, 4, 5, 6, 10	0, 1, 3, 4, 5, 6, 10	0, 1, 3, 4, 5, 6, 10
RAID Rebuild™ support		•	•	
Enterprise expert support		•	•	•



Geometry Example





Four types of delay

1. Rotational Delay
 - Time to rotate the desired sector to the read head
 - Related to RPM
2. Seek delay
 - Time to move the read head to a different track
3. Transfer time
 - Time to read or write bytes
4. Controller Overhead
 - Overhead for the request management



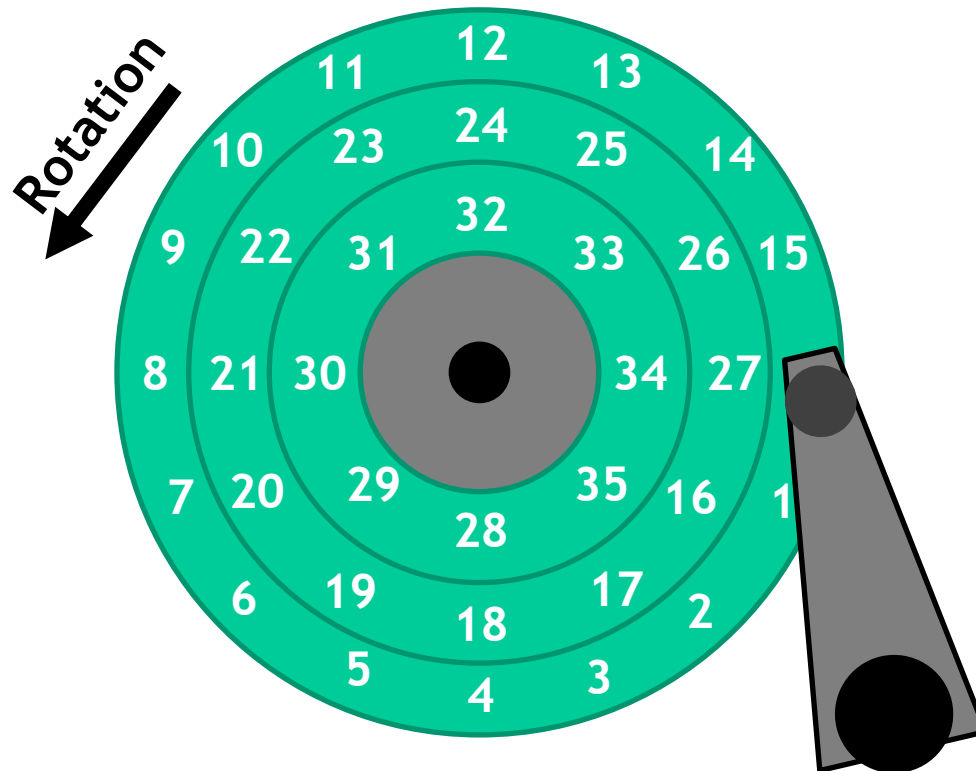
Rotational Delay



Full rotation delay is $R = 1/\text{DiskRPM}$

- In seconds $R_{\text{sec}} = 60 \cdot R$

$$T_{\text{rotation_AVG}} = R_{\text{sec}} / 2$$





Seek Time



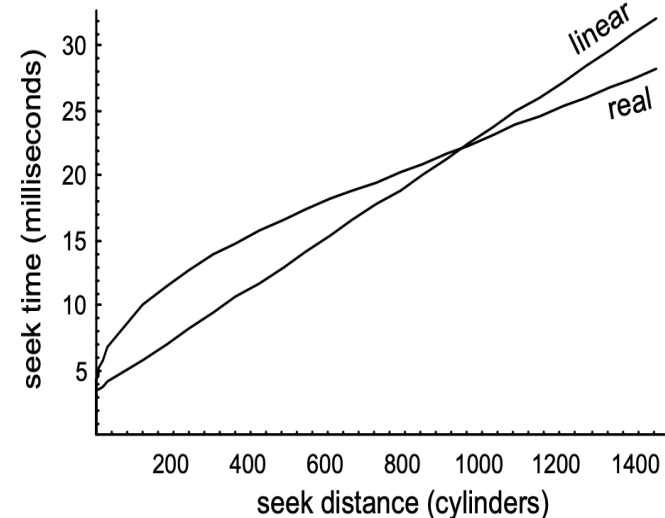
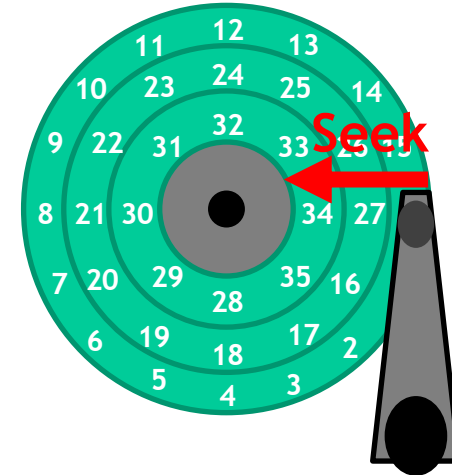
Time to move the head to a different track

Several Phases:

- Acceleration
- Coasting (constant speed)
- Deceleration
- Settling

T_{seek} modeling consider a linear dependency with the distance

$$T_{seek_AVG} = T_{seek_MAX} / 3$$





Transfer Time and Controller Overhead



Transfer time

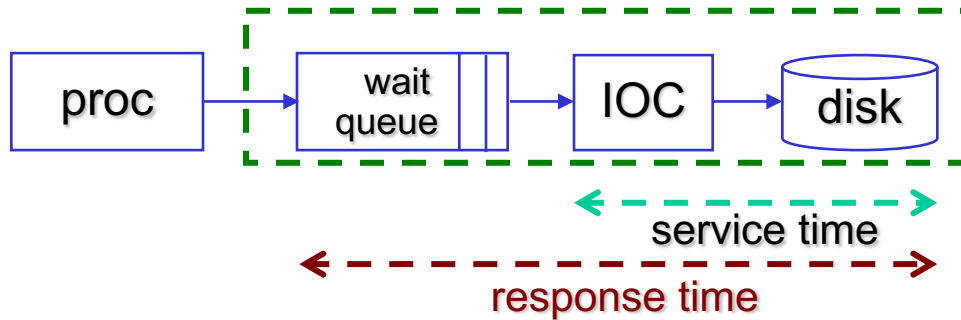
- Final phase of the I/O that takes place
- Time that consider that data is either read from or written to the surface
- Includes the time for the head to pass on the sectors and the I/O transfer
 - rotation speed, storing density

Controller Overhead

- Buffer management (data transfer) and interrupt sending time



How To Calculate Transfer Time



Service Time

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

Response time (\tilde{R})

- T_{queue} (waiting for the resource) + $T_{I/O}$
- Where T_{queue} depends on
 - queue-length, resource utilization, mean and variance of disk service time (distribution) and request arrival distribution



Exercise 1: mean service time of a I/O operation

read/write of a sector of 512 Byte = 0.5 KB

- data **transfer** rate: 50 MB/sec
- **rotation** speed: 10000 RPM (round per minute)
- mean **seek** time: 6ms
- overhead **controller**: 0.2ms

Transfer time

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



Exercise 1: mean service time of a I/O operation



read/write of a sector of 512 Byte = 0.5 KB

- data **transfer** rate: 50 MB/sec
- **rotation** speed: 10000 RPM (round per minute)
- mean **seek** time: 6ms
- overhead **controller**: 0.2ms

mean latency: $(60\text{s/min}) \times 1000 / (2 \times 10000 \text{ rpm}) = 3.0\text{ms}$ (time for $\frac{1}{2}$ round)

transfer time: $(0.5\text{KB}) \times 1000 / (50 \times 1024\text{KB/s}) = 0.01\text{ms}$

mean I/O service time = $\begin{matrix} \text{seek} & \text{latency} & \text{transfer} & \text{controller} \\ 6\text{ms} & + & 3\text{ms} & + & 0.01\text{ms} & + & 0.2\text{ms} & = & 9.21\text{ms} \end{matrix}$



The previous service times considers only the very pessimistic case where sectors are fragmented on the disk in worst possible way

- Files are very small (each file is contained in one block)
- or the disk is very (externally) fragmented

Thus, each access to a sector requires to pay

- Rotational latency, and
- Seek time

In many circumstances, this is not the case:

- files are larger than one block, and
- they are stored in a contiguous way

We can measure the *data locality* of a disk as the percentage of blocks that do not need seek or rotational latency to be found.



Exercise 2: data locality (see the values of exercise 1)



Calculate the average time for read/write a sector of 512 Byte = 0.5 KB

- **CONSIDERING A DATA LOCALITY = 75%**
- data **transfer** rate: 50 MB/sec
- **rotation** speed: 10000 RPM (round per minute)
- mean **seek** time: 6ms
- overhead **controller**: 0.2ms

Average transfer time

$$T_{I/O} = (1-DL) * (T_{seek} + T_{rotation}) + T_{transfer} + T_{controller}$$



Exercise 2: data locality (see the values of exercise 1)



Calculate the average time for read/write a sector of 512 Byte = 0.5 KB

- **CONSIDERING A DATA LOCALITY = 75%**
 - data **transfer** rate: 50 MB/sec
 - **rotation** speed: 10000 RPM (round per minute)
 - mean **seek** time: 6ms
 - overhead **controller**: 0.2ms
-
- **data locality l=75%**: seek+RotLatency affect only **25%** of the operations
 - $T = (1 - l) * (T_s + T_{rl}) + T_c + T_t$
- $(6.0 + (0.5 \times 60 \times 10^3 / 10000)) \times 0.25 + (0.5 \text{ KB} \times 1000 / (50\text{MB} \times 2^{10})) + 0.2$
- mean time of one I/O op. = $(0.25 \times (6+3)) + 0.01 + 0.2 = 2.46 \text{ ms}$



Exercise 3: influence of “not optimal” data allocation



time to transfer a file of 1MB

(10 blocks of 1/10 MB “not well” distributed on disk) for each block
(values as in ex.1)

Case A (locality = 100%):

- 1 initial seek: 6 ms
- 1 tot. latency: 3 ms
- 1 global transfer 1 MB: $(1/50) \times 1000 = 20$ ms
- **total time: 29 ms**

Case B (locality = 0%):

- 1 seek: 6 ms
- 1 rot. latency: 3 ms
- 1 partial transfer (1/10): 2 ms
- **total time: $(6 + 3 + 2) \times 10 = 110$ ms (+ 279% !)**

(controller times not considered)



- Caching helps improve disk performance
- But it can't make up for poor random access times
- Key idea:
 - If there are a queue of requests to the disk, they can be reordered to improve performance
 - Estimation of the request length is feasible knowing the position on the disk of the data
 - Several scheduling algorithms
 - First come, first serve (FCFC)
 - Shortest seek time first (SSTF)
 - SCAN, otherwise know as the elevator algorithm
 - C-SCAN, C-LOOK, etc.

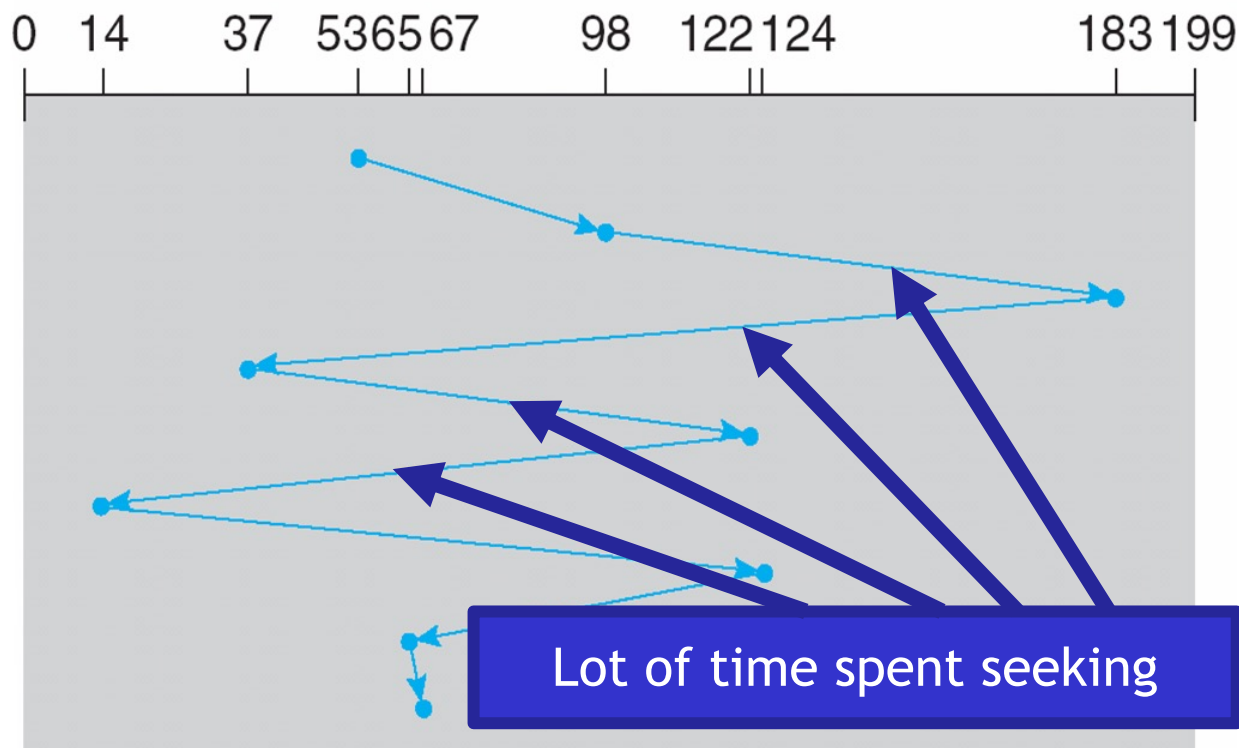


- Most basic scheduler, serve requests in order

- Head starts at block 53

- Queue:

- 98,
- 183,
- 37,
- 122,
- 14,
- 124,
- 65,
- 67



- Total movement: 640 cylinders

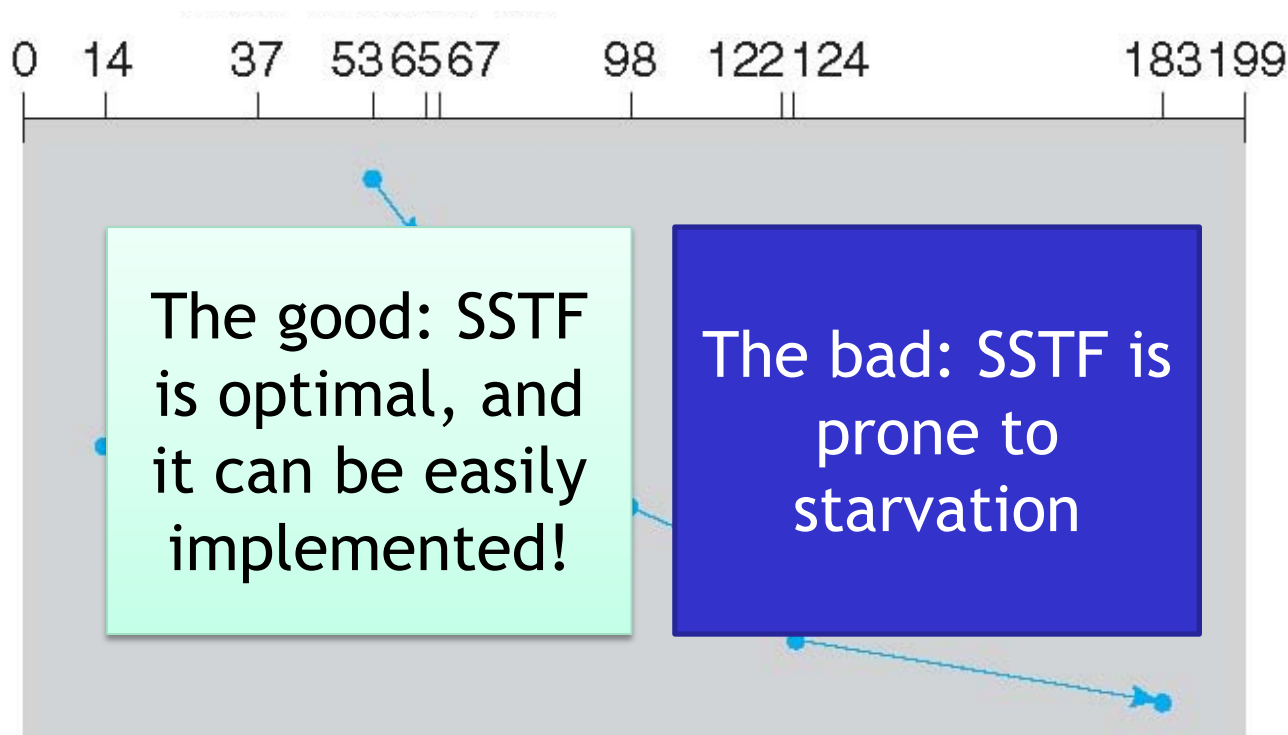


- Idea: minimize seek time by always selecting the block with the shortest seek time

- Head starts at block 53

- Queue:

- 98,
- 183,
- 37,
- 122,
- 14,
- 124,
- 65,
- 67



- Total movement: 236 cylinders



SCAN Example

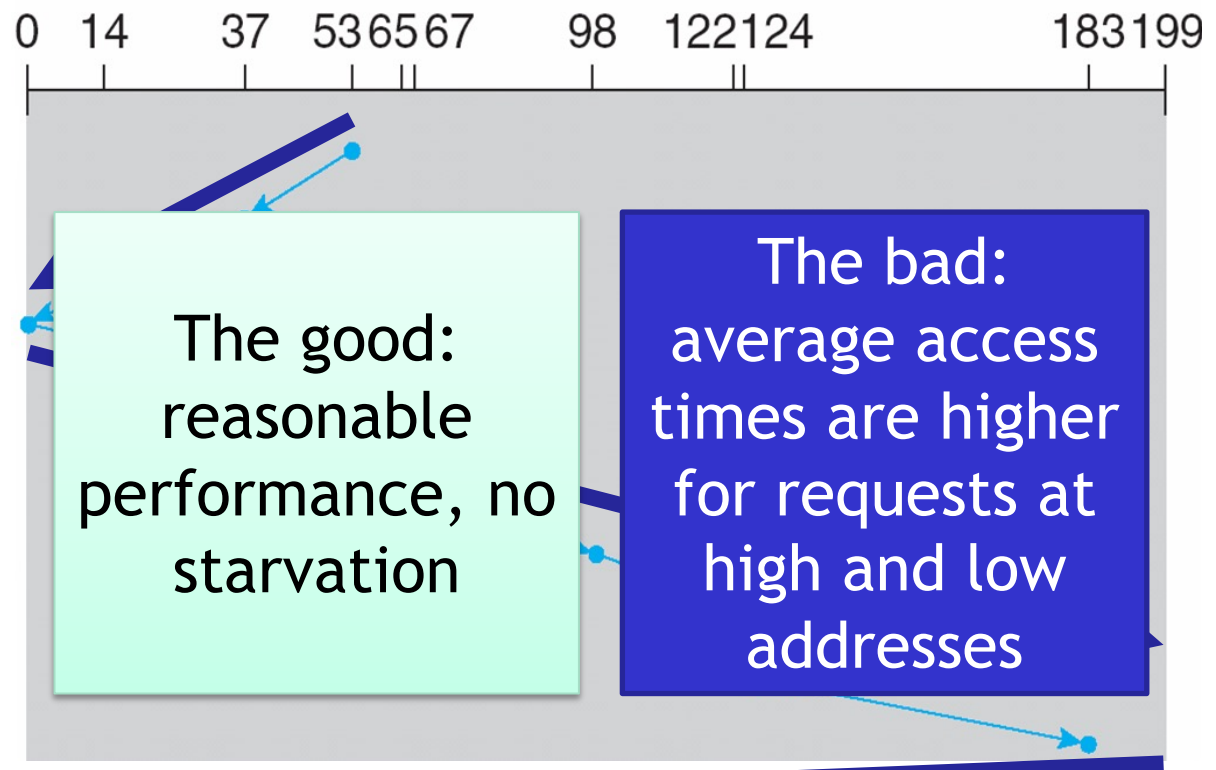


- Head **sweeps** across the disk servicing requests in order

- Head starts at block 53

- Queue:

- 98,
- 183,
- 37,
- 122,
- 14,
- 124,
- 65,
- 67



- Total movement: 236 cylinders



C-SCAN Example

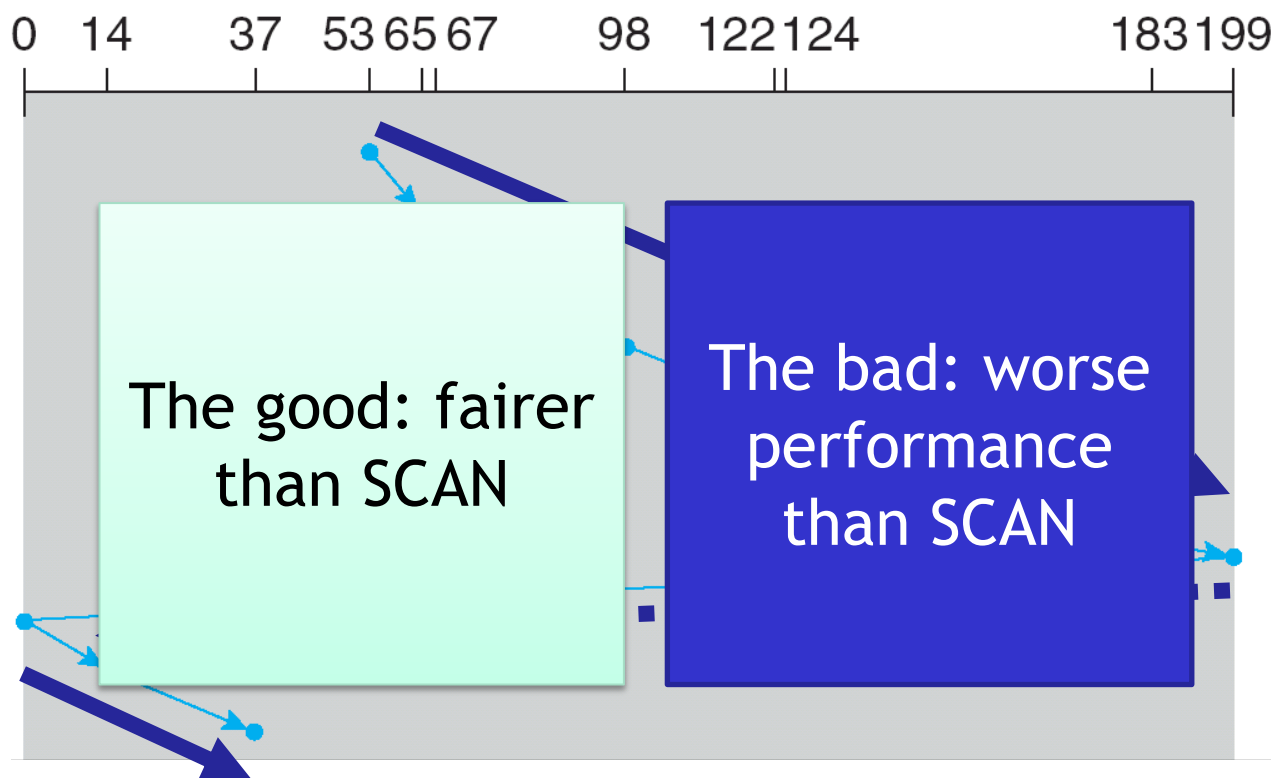


- Like SCAN, but only service requests in one direction (Circular SCAN)

- Head starts at block 53

- Queue:

- 98,
- 183,
- 37,
- 122,
- 14,
- 124,
- 65,
- 67



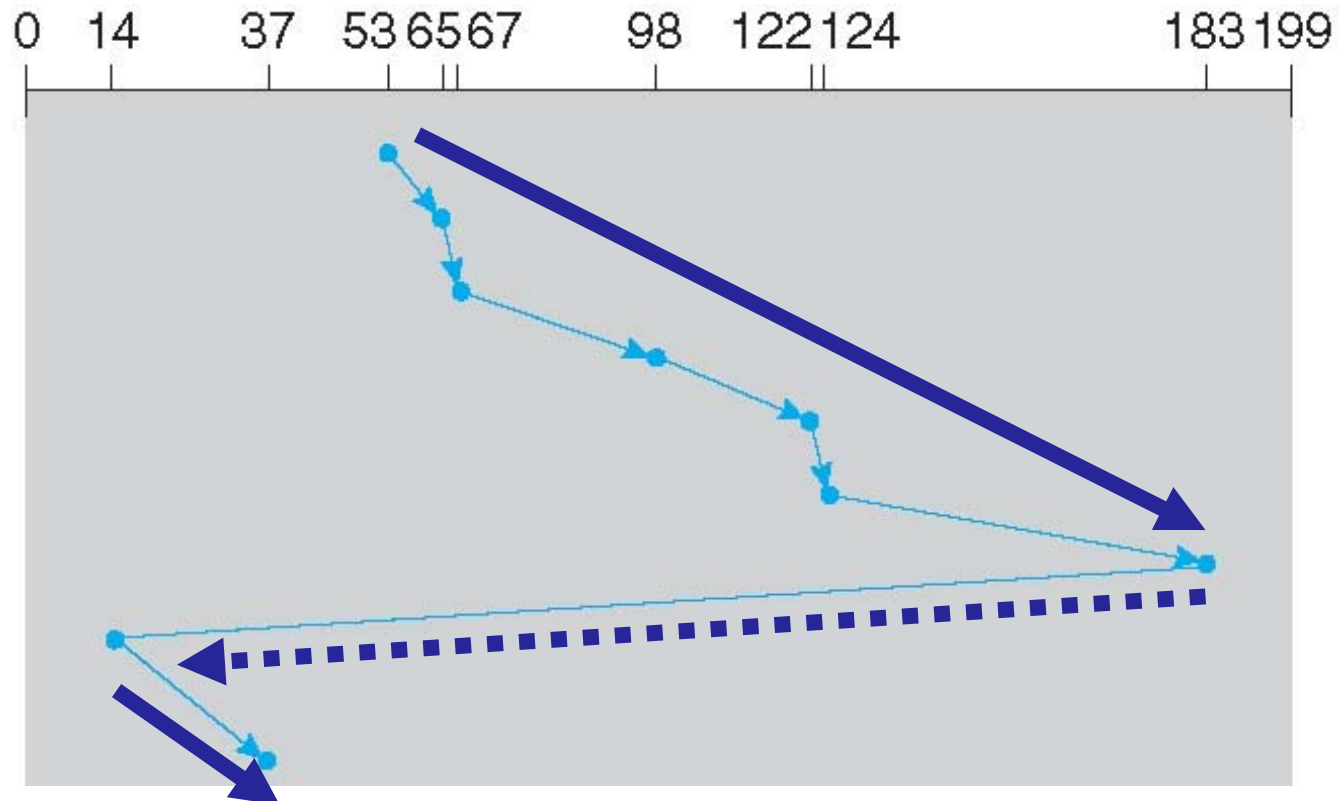
- Total movement: 382 cylinders



C-LOOK Example



- C-SCAN Variant that peeks at the upcoming addresses in the queue
 - Head only goes as far as the last request
- Head starts at block 53
- Queue:
 - 98,
 - 183,
 - 37,
 - 122,
 - 14,
 - 124,
 - 65,
 - 67



- Total movement: 322 cylinders