Mass Storage Structures

Mass-Storage Structure

The file system can be viewed logically as consisting of three parts.

File system interface	The user and programmer interface to the file system.
File system implementation	The internal data structures and algorithms used by the O.S. to implement the above interface.
Mass-storage structure	The secondary and tertiary storage structures.

Mass-Storage Structure

The secondary and tertiary storage structures are the lowest level of the file system.

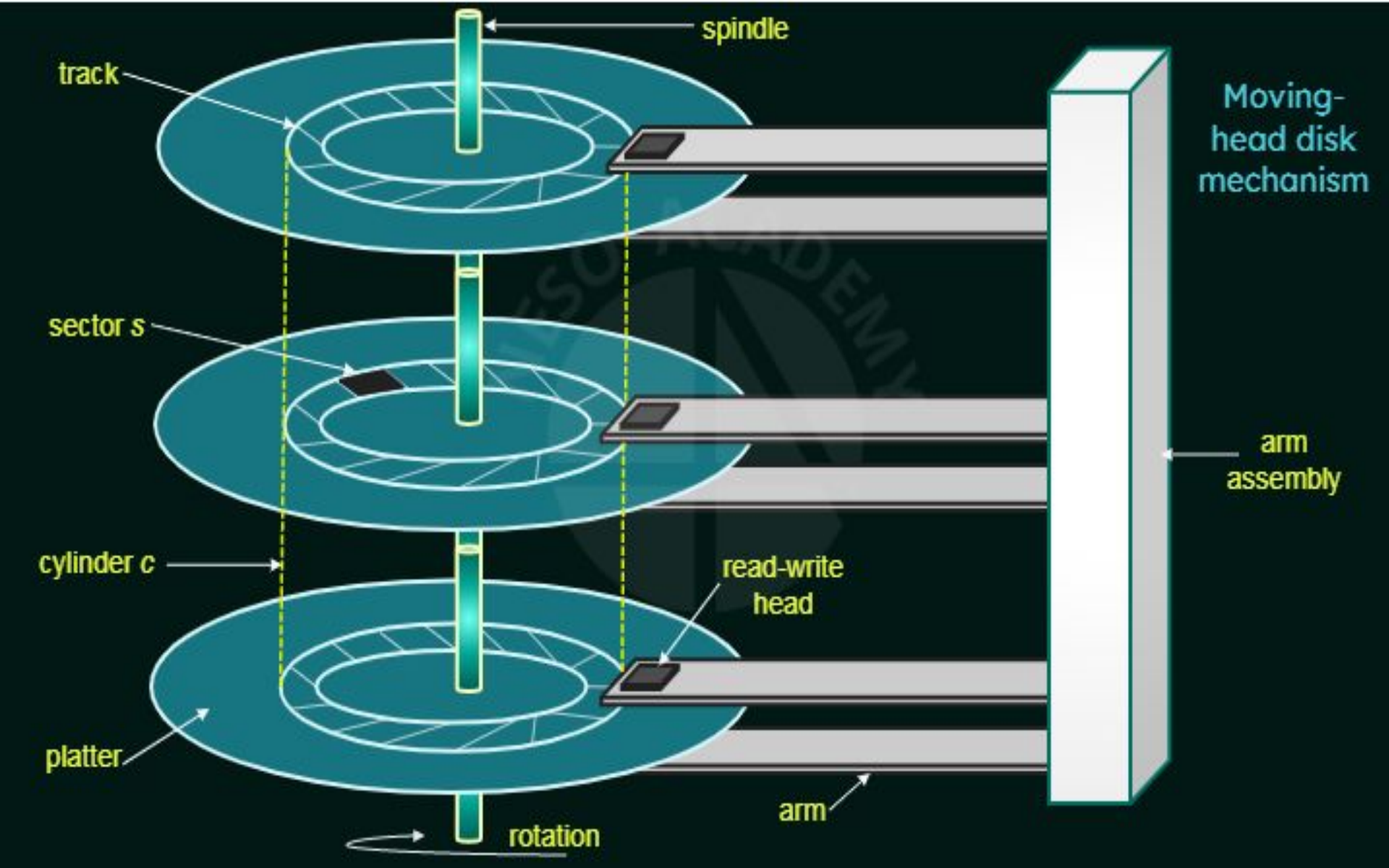
Topics to be discussed:

- The physical structure of magnetic disks and magnetic tapes.
- Disk-scheduling algorithms.
- Disk formatting and management of boot blocks, damaged blocks, and swap space.

Magnetic Disks

Magnetic disks provide the bulk of secondary storage for modern computer systems.

- Each disk platter has a flat circular shape, like a CD.
- Common platter diameters range from 1.8 to 5.25 inches.
- The two surfaces of a platter are covered with a magnetic material.
- We store information by recording it magnetically on the platters.
- A read-write head flies just above each surface of every platter.
- The heads are attached to a disk arm that moves all the heads as a unit.


spindle
track
Moving-
head disk
mechanism
sector s
arm
assembly
cylinder c
read-write
head
platter
arm
rotation

Different parts

- The surface of a platter is logically divided into circular tracks.
- The circular tracks are subdivided into sectors.
- The set of tracks that are at one arm position makes up a cylinder.
- There may be thousands of concentric cylinders in a disk drive, and each track may contain hundreds of sectors.
- The storage capacity of common disk drives is measured in gigabytes.

Disk speeds

When the disk is in use, a drive motor spins it at high speed. Most drives rotate 60 to 200 times per second.

Transfer rate:

The rate at which data flow between the drive and the computer.

Positioning time/ Random access time:

The time to move the disk arm to the desired cylinder (seek time) + the time for the desired sector to rotate to the disk head (rotational latency).

Typical disks can transfer several megabytes of data per second, and they have seek times and rotational latencies of several milliseconds.

Head crash

- The disk head flies on an extremely thin cushion of air (measured in microns).
- There is a danger that the head will make contact with the disk surface.
- Although the disk platters are coated with a thin protective layer, sometimes the head will damage the magnetic surface.
- This accident is called a head crash.
- A head crash normally cannot be repaired; the entire disk must be replaced.

Attachments

A disk drive is attached to a computer by a set of wires called an I/O bus.

Some buses include:

- Enhanced integrated drive electronics (EIDE)
- Advanced technology attachment (ATA)
- Serial ATA (SATA)
- Universal serial bus (USB)
- Fiber channel (FC)
- Small Computer System Interface (SCSI) buses

The data transfers on a bus are carried out by special electronic processors called controllers.

Magnetic Tapes

Magnetic tape was used as an early secondary-storage medium.

It is relatively permanent and can hold large quantities of data, its access time is slow compared with that of main memory and magnetic disk.

Random access to magnetic tape is about a thousand times slower than random access to magnetic disk, so tapes are not very useful for secondary storage.

Why?

- A tape is kept in a spool and is wound or rewound past a read-write head.
- Moving to the correct spot on a tape can take minutes.
- Once positioned, tape drives can write data at speeds comparable to disk drives.
- Tape capacities vary greatly, depending on the particular kind of tape drive.

  Typically, they store from 20 GB to 200 GB.
- Tape categories:

  Based on width: 4, 8, and 19 millimeters and 1/4 and 1/2 inch.

  Based on technology: LTO-2 and SDLT.

Tapes are used mainly for backup, for storage of infrequently used information, and as a medium for transferring information from one system to another.

Disk Structure

Modern disk drives are addressed as large one-dimensional arrays of logical blocks.

↓

the smallest unit of transfer

↓

usually 512 bytes

- The one-dimensional array of logical blocks is mapped onto the sectors of the disk sequentially.
- Sector 0 is the first sector of the first track on the outermost cylinder.
- The mapping proceeds in order through that track, then through the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.

By using this mapping, we can:

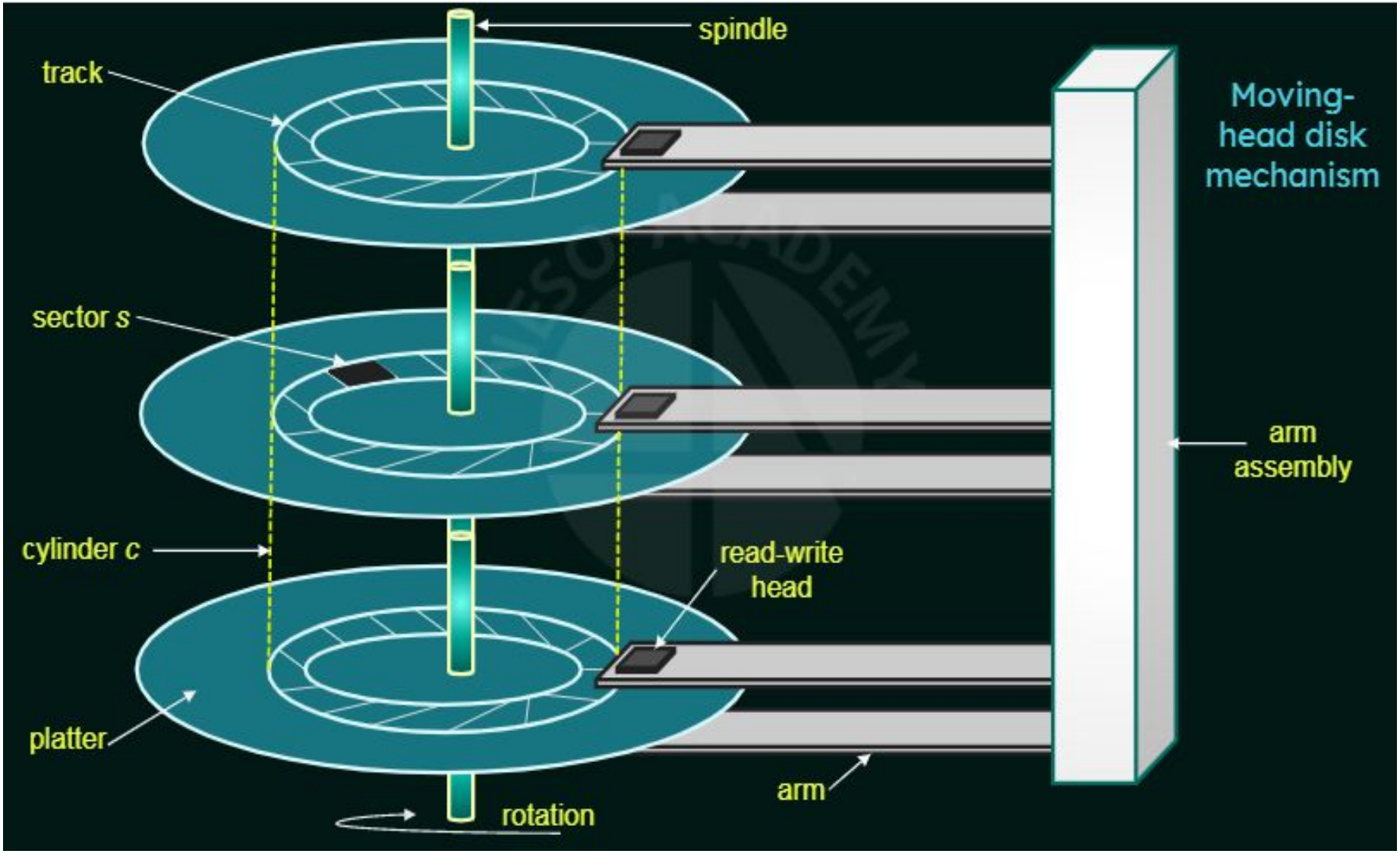
Convert a logical block number into an old-style disk address that consists of a cylinder number, a track number within that cylinder, and a sector number within that track.

But in practice, it is difficult to perform, this translation.

WHY ?

1. Most disks have some defective sectors, but the mapping hides this by substituting spare sectors from elsewhere on the disk.

2. The number of sectors per track is not a constant on some drives.


spindle
track
Moving-
head disk
mechanism
sector s
arm
assembly
cylinder c
read-write
head
platter
arm
rotation

Constant Linear Velocity (CLV)

- On media that use CLV, the density of bits per track is uniform.
- The farther a track is from the center of the disk, the greater its length, so the more sectors it can hold.
- As we move from outer zones to inner zones, the number of sectors per track decreases.
- Tracks in the outermost zone typically hold 40 % more sectors than tracks in the innermost zone.
- The drive increases its rotation speed as the head moves from the outer to the inner tracks to keep the same rate of data moving under the head.
- This method is used in CD-ROM and DVD-ROM drives.

Constant Angular Velocity (CAV)

- The disk rotation speed can stay constant.
- The density of bits decreases from inner tracks to outer tracks to keep the data rate constant.

The number of sectors per track has been increasing as disk technology improves, and the outer zone of a disk usually has several hundred sectors per track.

Similarly, the number of cylinders per disk has been increasing; large disks have tens of thousands of cylinders.

Disk Attachment

Disk storage can be accessed by computers in two ways:

1. Via I/O ports (Host-Attached Storage)
2. Via a remote host in a distributed file system (Network-Attached Storage)

We will discuss about:

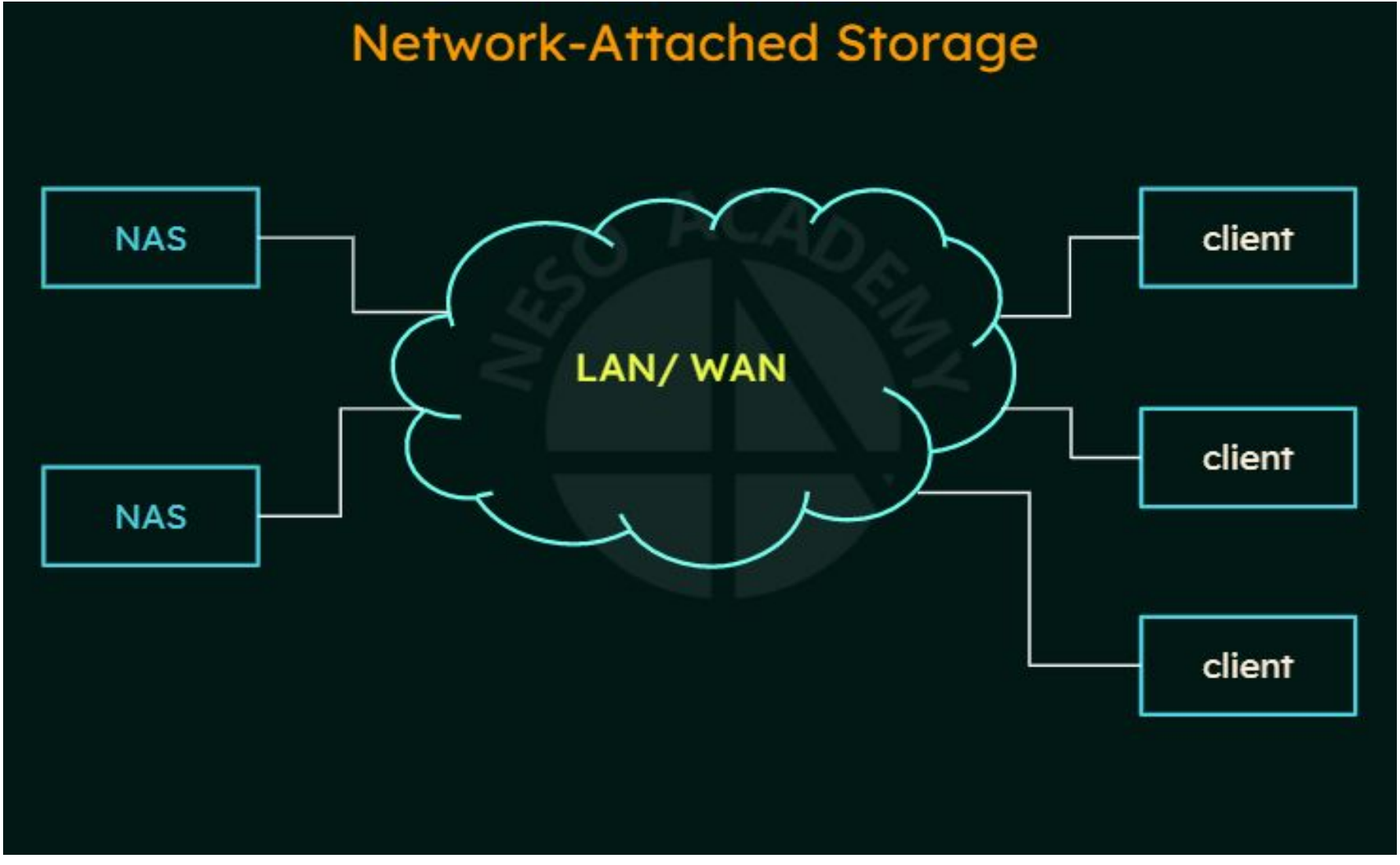
1. Host-Attached Storage
2. Network-Attached Storage
3. Storage-Area Network

Host-Attached Storage

- Host-attached storage is storage accessed through local I/O ports (which use several technologies).
- The typical desktop PC uses an I/O bus architecture called:
  IDE (Integrated Drive Electronics) or ATA (Advanced Technology Attachment).
  - Supports a maximum of two drives per I/O bus.
- A newer, similar protocol that has simplified cabling is SATA (Serial ATA).
- High-end workstations and servers use more sophisticated I/O architectures, such as:
  SCSI (Small Computer System Interface) and Fiber Channel (FC)
- The SCSI protocol supports a maximum of 16 devices on the bus.
  Physical medium → usually a ribbon cable having a large number of conductors.
- FC → high-speed serial architecture that can operate over optical fiber or over a four-conductor copper cable. PC variant → arbitrated loop (FC-AL) → can address 126 devices.

Network-Attached Storage

- A network-attached storage (NAS) device is a special-purpose storage system that is accessed remotely over a data network.
- Clients access network-attached storage via a remote-procedure-call interface such as NFS for UNIX systems or CIFS for Windows machines.
- The remote procedure calls (RPCs) are carried via TCP or UDP over an IP network-usually the same local-area network (LAN) that carries all data traffic to the clients.
- The network attached storage unit is usually implemented as a RAID array with software that implements the RPC interface.

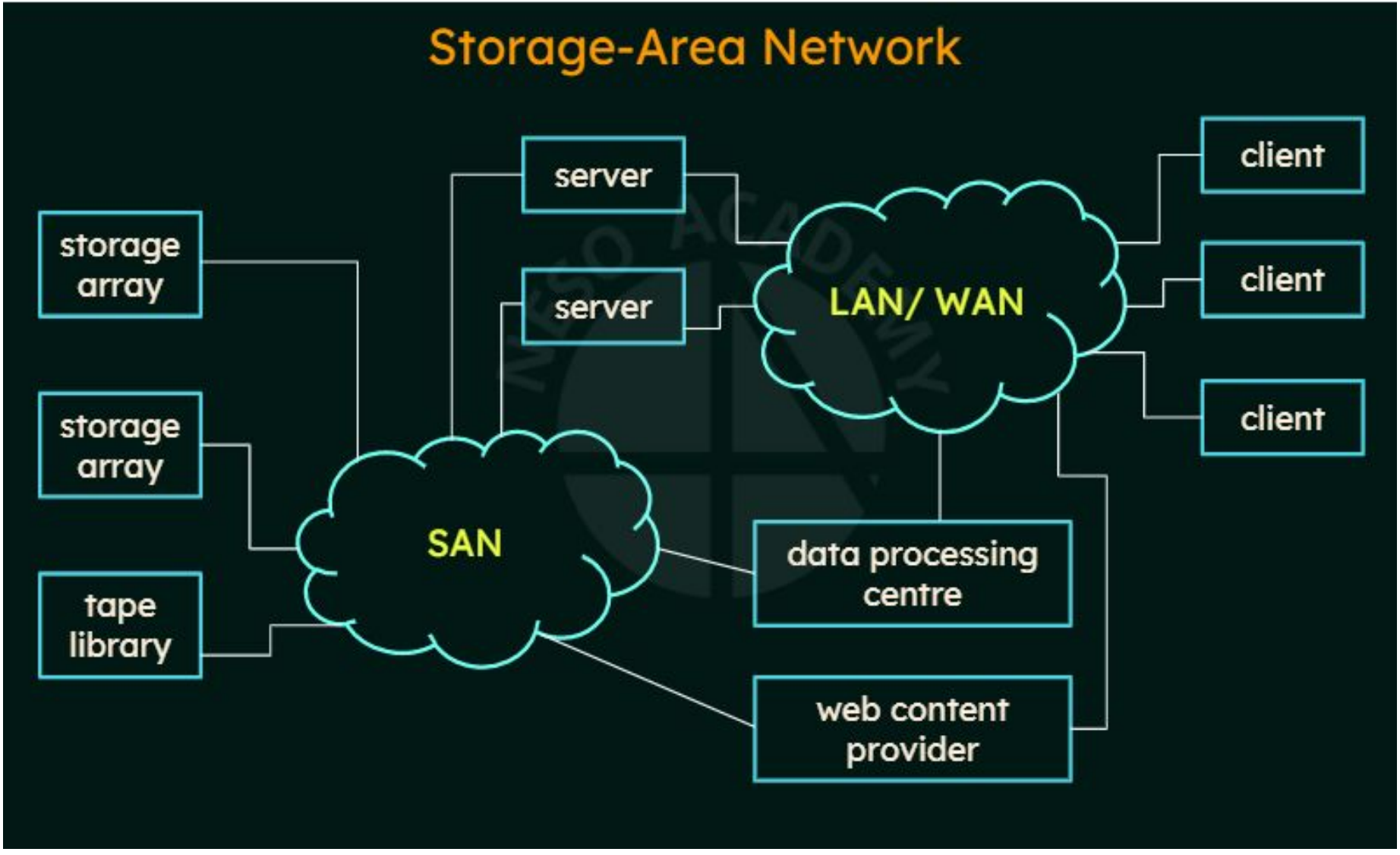

Network-Attached Storage
NAS
NAS
LAN/ WAN
client
client
client

Storage-Area Network

Drawback of network-attached storage systems:

The storage I/O operations consume bandwidth on the data network, thereby increasing the latency of network communication.

- A storage-area network (SAN) is a private network (using storage protocols rather than networking protocols) connecting servers and storage units.
- Multiple hosts and multiple storage arrays can attach to the same SAN, and storage can be dynamically allocated to hosts.
- E.g. If a host is running low on disk space, the SAN can be configured to allocate more storage to that host.
- SANs make it possible for clusters of servers to share the same storage and for storage arrays to include multiple direct host connections.


Storage-Area Network
server
server
client
client
client
storage array
storage array
tape library
LAN/ WAN
SAN
data processing centre
web content provider

Disk Scheduling

One of the responsibilities of the operating system is to use the hardware efficiently.

To achieve this, the disks need to have:

- Fast access time.
- Large disk bandwidth.

Access Time

Seek Time + Rotational Latency

Seek Time: The time for the disk arm to move the heads to the cylinder containing the desired sector.

Rotational Latency: The additional time for the disk to rotate the desired sector to the disk head.

Disk Scheduling

One of the responsibilities of the operating system is to use the hardware efficiently.

To achieve this, the disks need to have:

- Fast access time.
- Large disk bandwidth.

Disk Bandwidth

The total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.

$$\frac{Total\ number\ of\ bytes\ transferred}{Total\ time\ between\ the\ first\ request\ for\ service\ and\ the\ completion\ of\ the\ last\ transfer}$$

How can we improve the access time & bandwidth

We can improve both the access time and the bandwidth by scheduling the servicing of disk I/O requests in a good order.

Whenever a process needs I/O to or from the disk, it issues a system call to the OS.

The request specifies several pieces of information:

- Whether this operation is input or output.
- What the disk address for the transfer is.
- What the memory address for the transfer is.
- What the number of sectors to be transferred is.

If the desired disk drive and controller are available, the request can be serviced immediately.

If the drive or controller is busy, any new requests for service will be placed in the queue of pending requests for that drive.

How can we improve the access time & bandwidth

We can improve both the access time and the bandwidth by

scheduling the servicing of disk I/O requests in a good order.

Whenever a process needs I/O to or from the disk, it issues a system call to the OS.

The request specifies several pieces of information:

- Whether this operation is input or output.
- What the disk address for the transfer is.
- What the memory address for the transfer is.
- What the number of sectors to be transferred is.

If the desired disk drive and controller are available, the request can be serviced immediately.

If the drive or controller is busy, any new requests for service will be placed in the queue of pending requests for that drive. → Disk-scheduling algorithms.

Disk Scheduling Algorithms

First-Come, First-Served (FCFS)

FCFS is the simplest Disk Scheduling Algorithm.

Here the requests are addressed in the order they arrive in the disk queue.

Example:

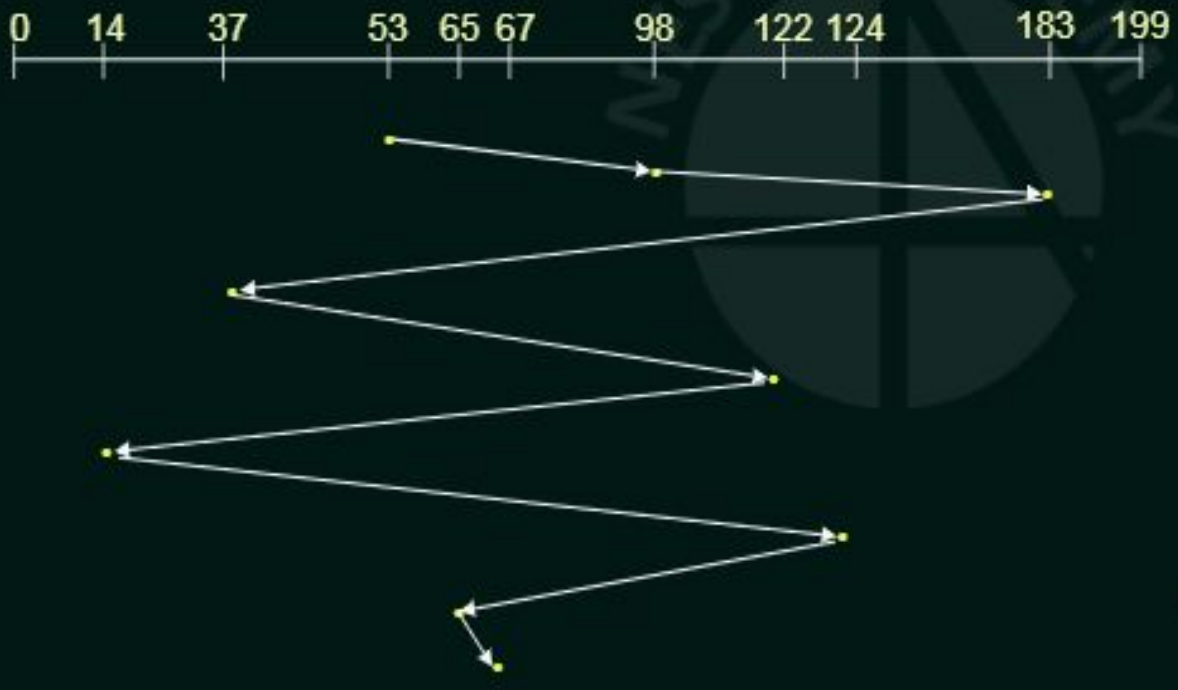
Consider a disk queue with requests for I/O to blocks on cylinders

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

FCFS disk scheduling

Queue: 98, 183, 37,122, 14, 124, 65, 67

Head starts at: 53



Total head movement

=

(98-53) + (183-98) +
(183-37) + (122-37) +
(122-14) + (124-14) +
(124-65) + (67-65)

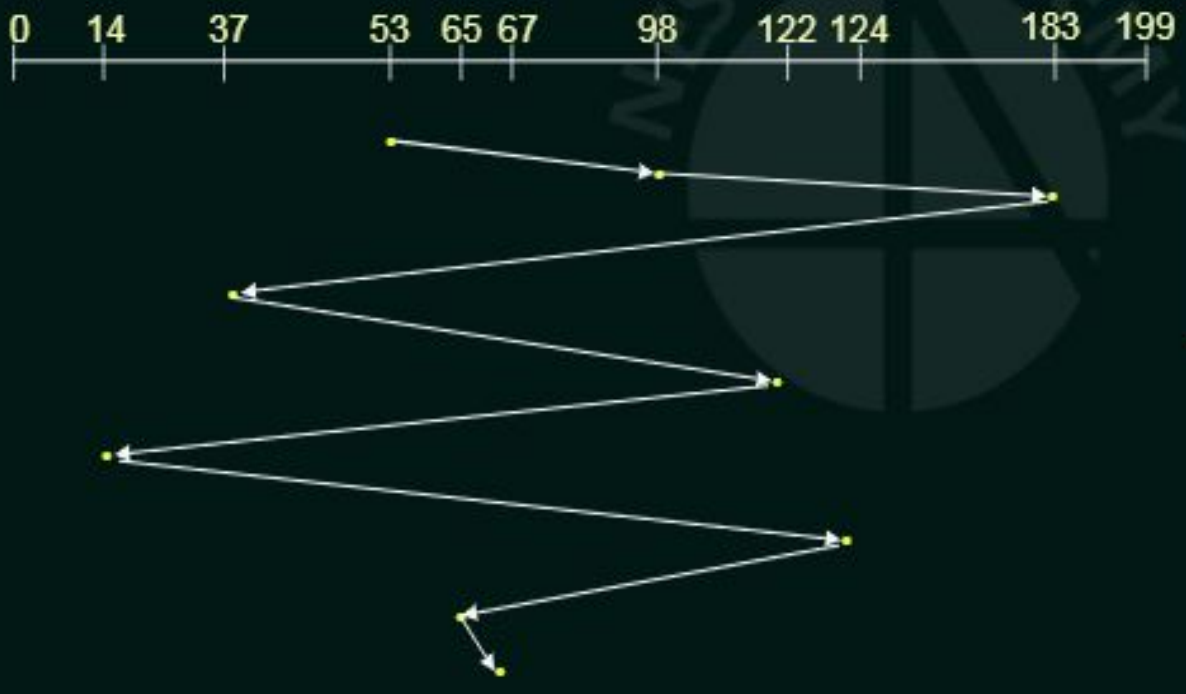
= 45 + 85 + 146 + 85 + 108 +
110 + 59 + 2

= 640

FCFS disk scheduling

Queue: 98, 183, 37,122, 14, 124, 65, 67

Head starts at: 53



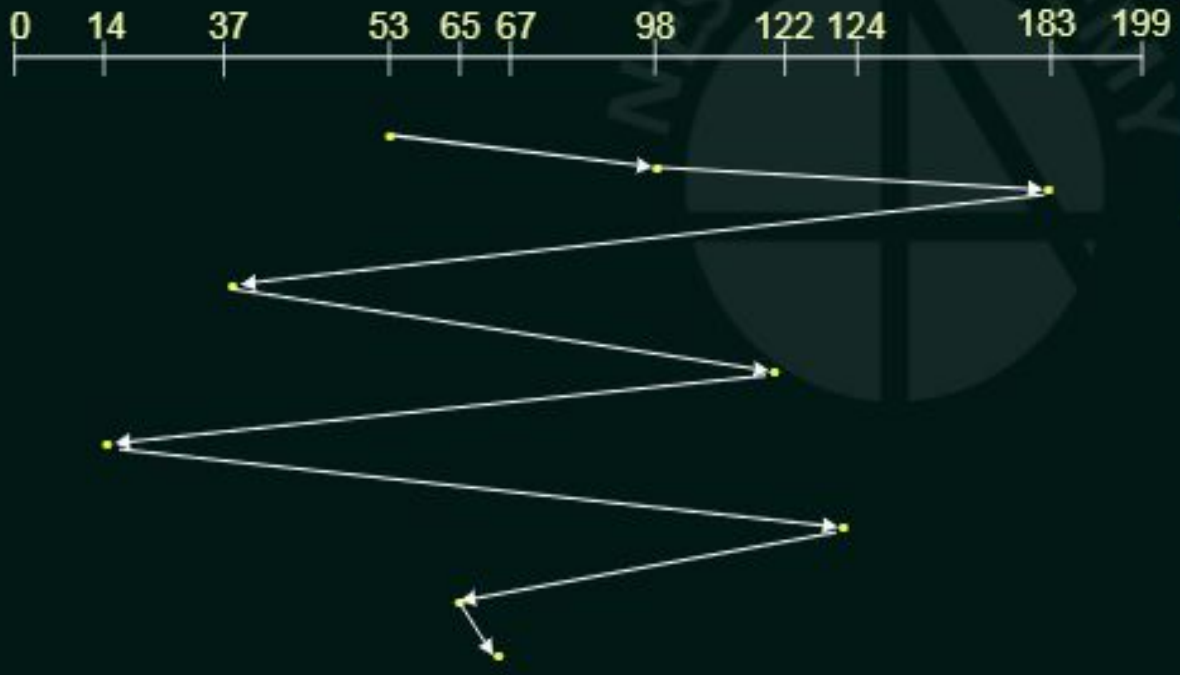
Advantages:

- The algorithm is intrinsically fair.
- No starvation.

FCFS disk scheduling

Queue: 98, 183, 37,122, 14, 124, 65, 67

Head starts at: 53



Disadvantages:

- Not the best performance.

Disk Scheduling Algorithms

Shortest-Seek-Time-First (SSTF)

It seems reasonable to service all the requests close to the current head position before moving the head far away to service other requests.

- The SSTF algorithm selects the request with the minimum seek time from the current head position.
- Since seek time increases with the number of cylinders traversed by the head, SSTF chooses the pending request closest to the current head position.

Example:

Consider a disk queue with requests for I/O to blocks on cylinders

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

SSTF disk scheduling

Queue: 98, 183, 37,122, 14, 124, 65, 67

Head starts at: 53

Total head movement

=

(65-53) + (67-65) +
(67-37) + (37-14) +
(98-14) + (122-98) +
(124-122) + (183-124)

= 12 + 2 + 30 + 23 + 84 + 24 + 2 + 59

= 236

SSTF disk scheduling

Queue: 98, 183, 37,122, 14, 124, 65, 67

Head starts at: 53

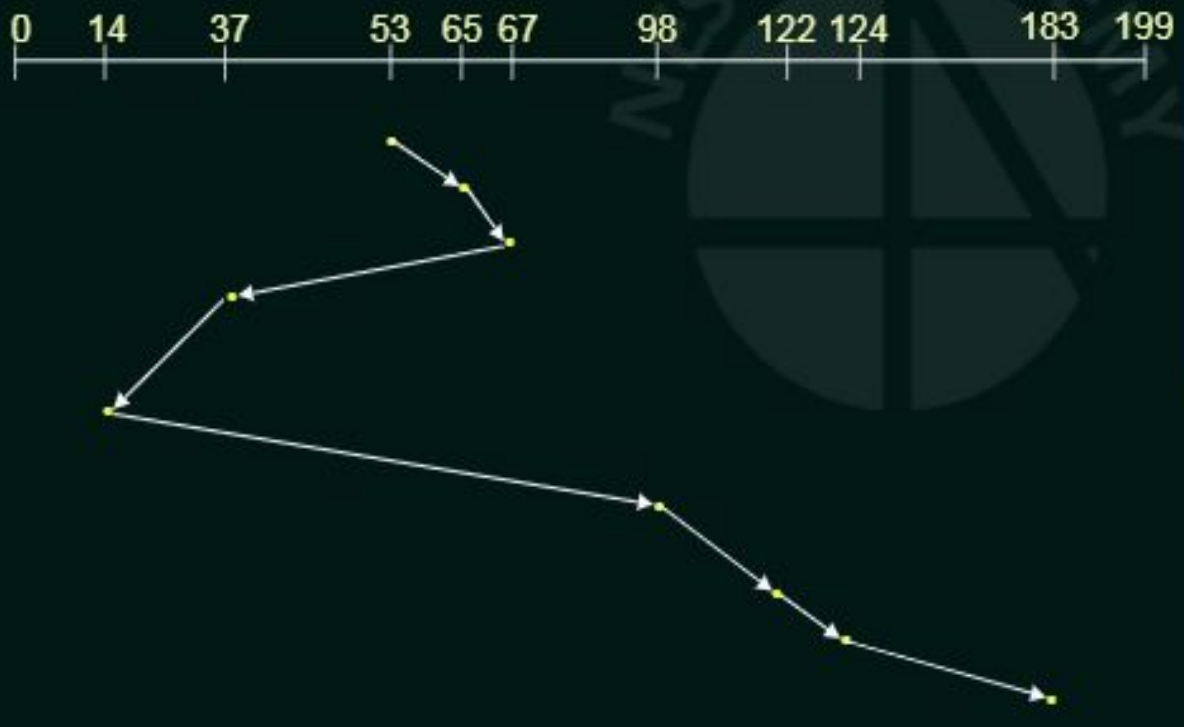
Advantages:

- This algorithm gives a substantial improvement in performance as compared to FCFS

SSTF disk scheduling

Queue: 98, 183, 37,122, 14, 124, 65, 67

Head starts at: 53



Disadvantages:

- Can cause starvation.
- Though it is better than FCFS, it is still not optimal.

Disk Scheduling Algorithms

SCAN (Elevator)

- In the SCAN algorithm, the disk arm starts at one end of the disk and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk.
- At the other end, the direction of head movement is reversed, and servicing continues.
- The head continuously scans back and forth across the disk.

Example:

Consider a disk queue with requests for I/O to blocks on cylinders

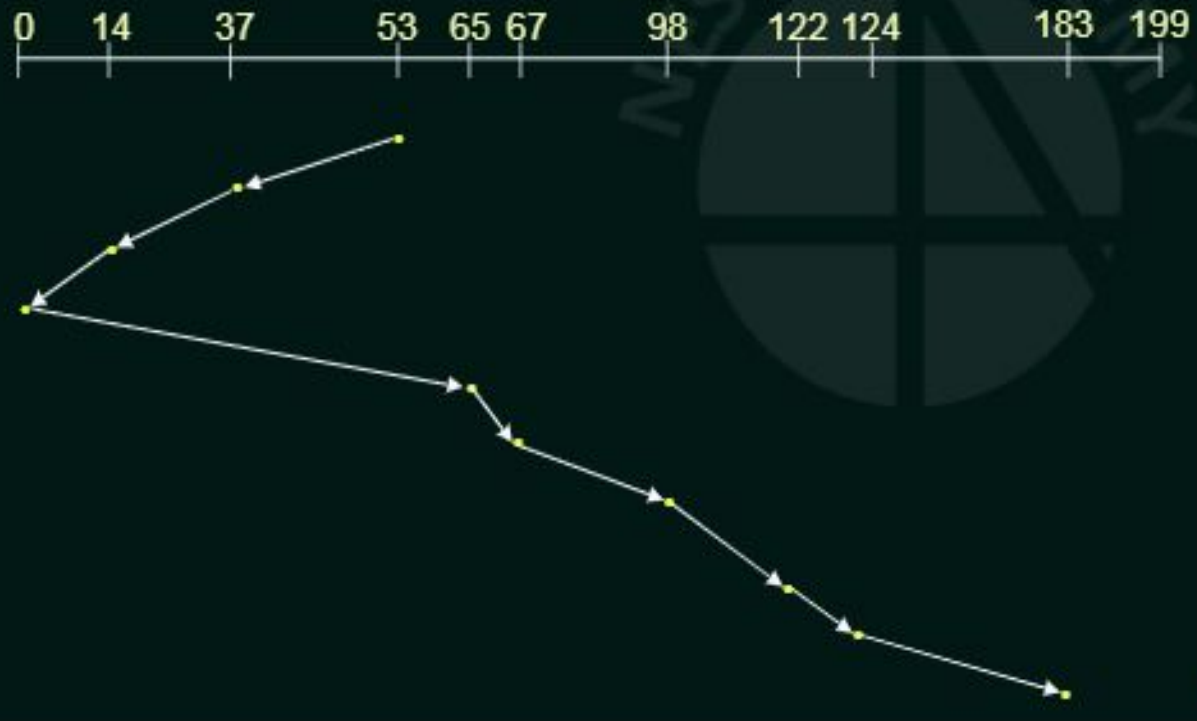
98, 183, 37,122, 14, 124, 65, 67

Let's assume that the direction of head movement is towards 0.

SCAN disk scheduling

Queue: 98, 183, 37,122, 14, 124, 65, 67

Head starts at: 53 (and direction of head movement is towards 0)



Total head movement

=

(53-37) + (37-14) +
(14-0) + (65-0) +
(67-65) + (98-67) +
(122-98) + (124-122) +
(183-124)

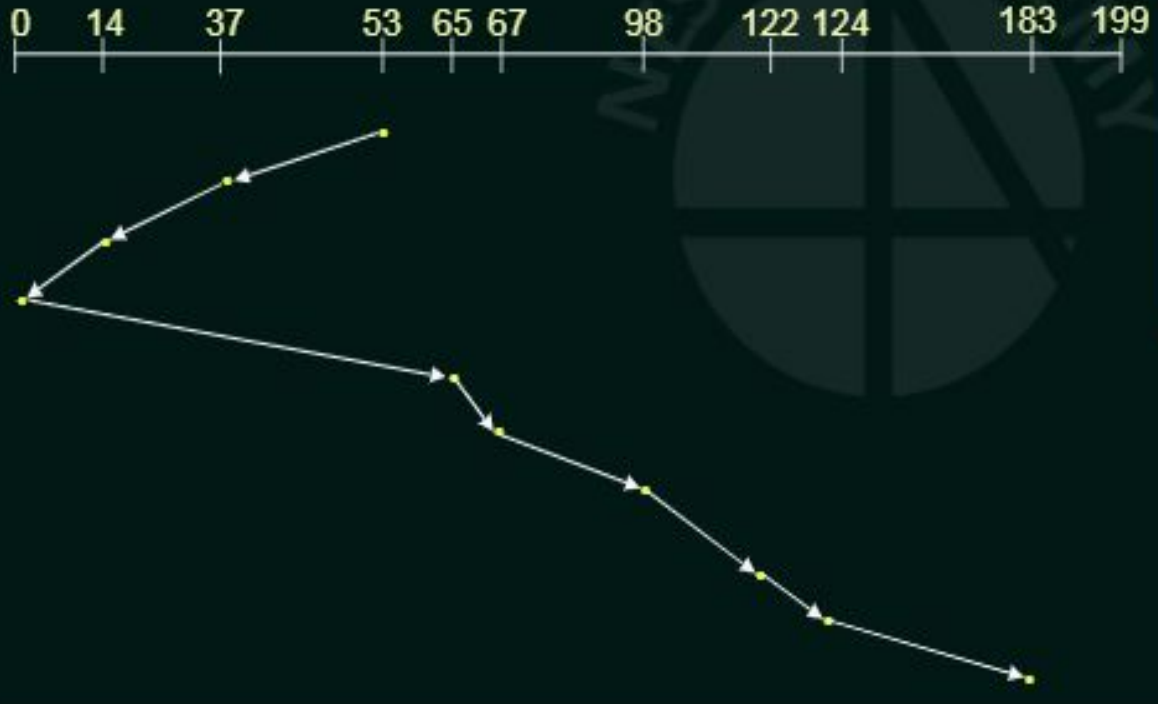
= 16 + 23 + 14 + 65 + 2 + 31 + 24 + 2 + 58

= 235

SCAN disk scheduling

Queue: 98, 183, 37,122, 14, 124, 65, 67

Head starts at: 53 (and direction of head movement is towards 0)



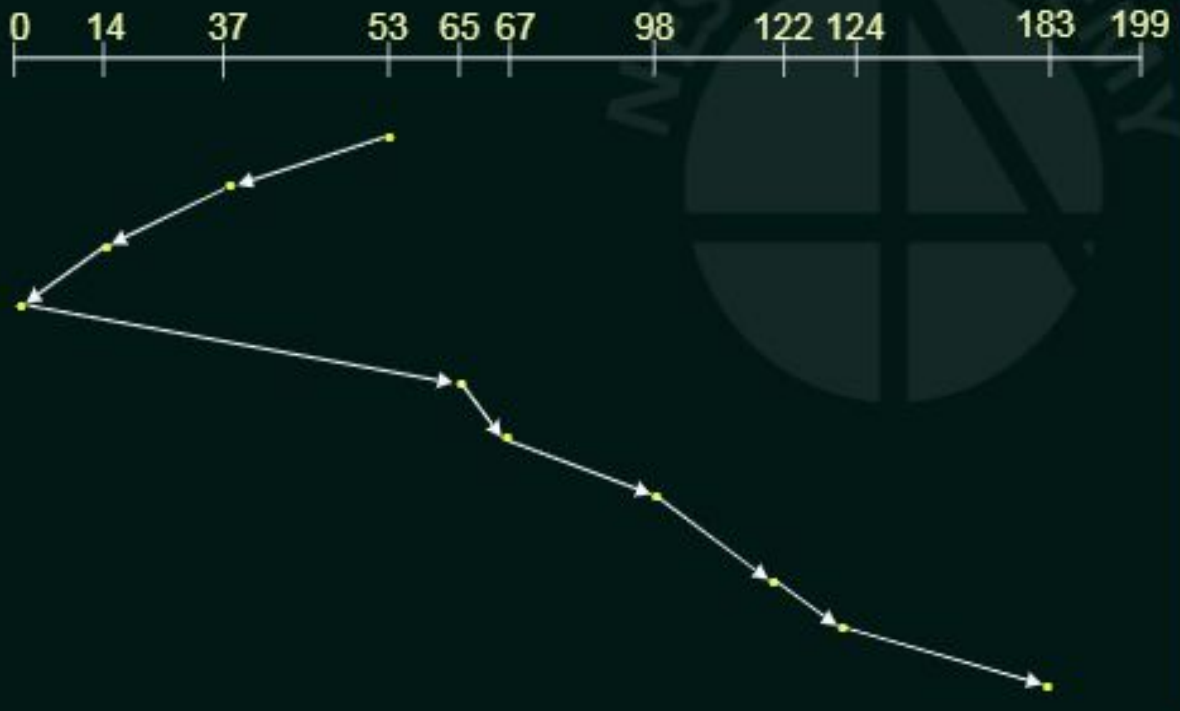
Advantages:

- This algorithm is better than FCFS.
- No starvation.

SCAN disk scheduling

Queue: 98, 183, 37,122, 14, 124, 65, 67

Head starts at: 53 (and direction of head movement is towards 0)



Disadvantages:

- If a request arrives in the queue just in front of the head, it will be serviced almost immediately; a request arriving just behind the head will have to wait until the arm moves to the end of the disk, reverses direction, and comes back.

Disk Scheduling Algorithms

Circular SCAN (C-SCAN)

- In SCAN algorithm, when the head reaches one end and reverses direction, the density of requests immediately in front of the head are relatively few as these cylinders have recently been serviced.
- The heaviest density of requests is at the other end of the disk. These requests have also waited the longest, so why not go there first?
- Like SCAN, C-SCAN moves the head from one end of the disk to the other, servicing requests along the way.
- When the head reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.

Example:

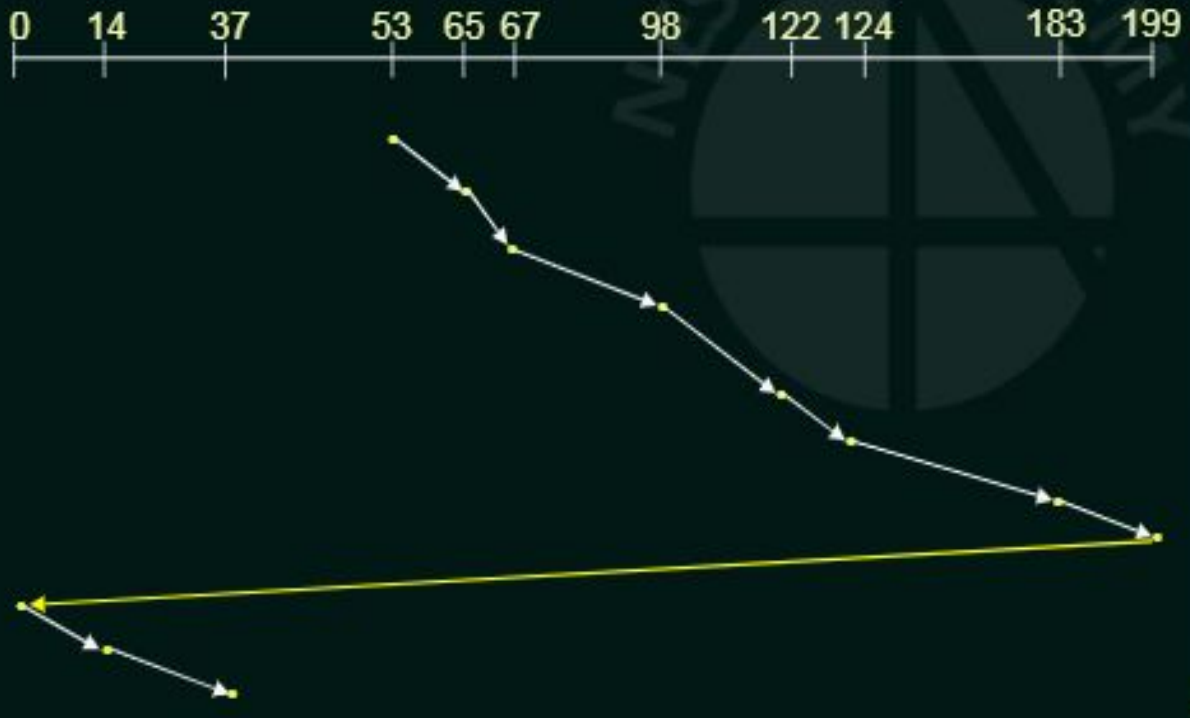
Consider a disk queue with requests for I/O to blocks on cylinders

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

C-SCAN disk scheduling

Queue: 98, 183, 37,122, 14, 124, 65, 67

Head starts at: 53 (and direction of head movement is towards 199)



Total head movement

=

(65-53) + (67-65) +
(98-67) + (122-98) +
(124-122) + (183-124) +
(199-183) + (199-0) +
(14-0) + (37-14)

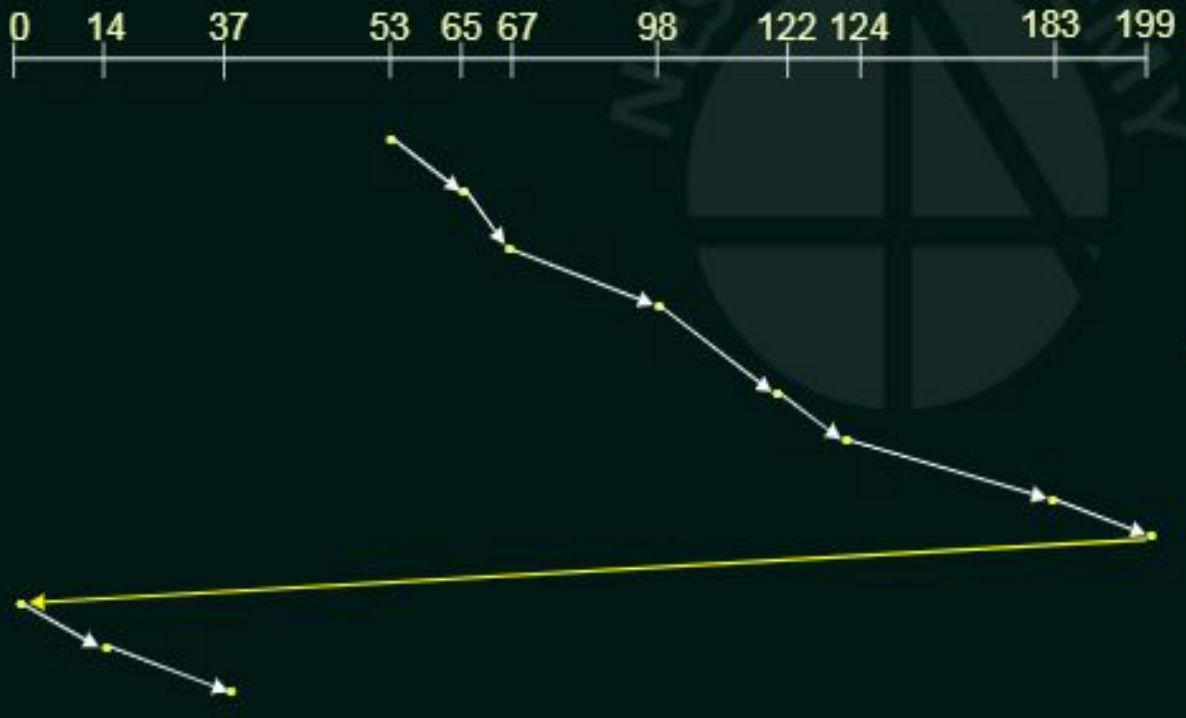
= 12 + 2 + 31 + 24 + 2 + 59 +
16 + 199 + 14 + 23

= 382

C-SCAN disk scheduling

Queue: 98, 183, 37,122, 14, 124, 65, 67

Head starts at: 53 (and direction of head movement is towards 199)



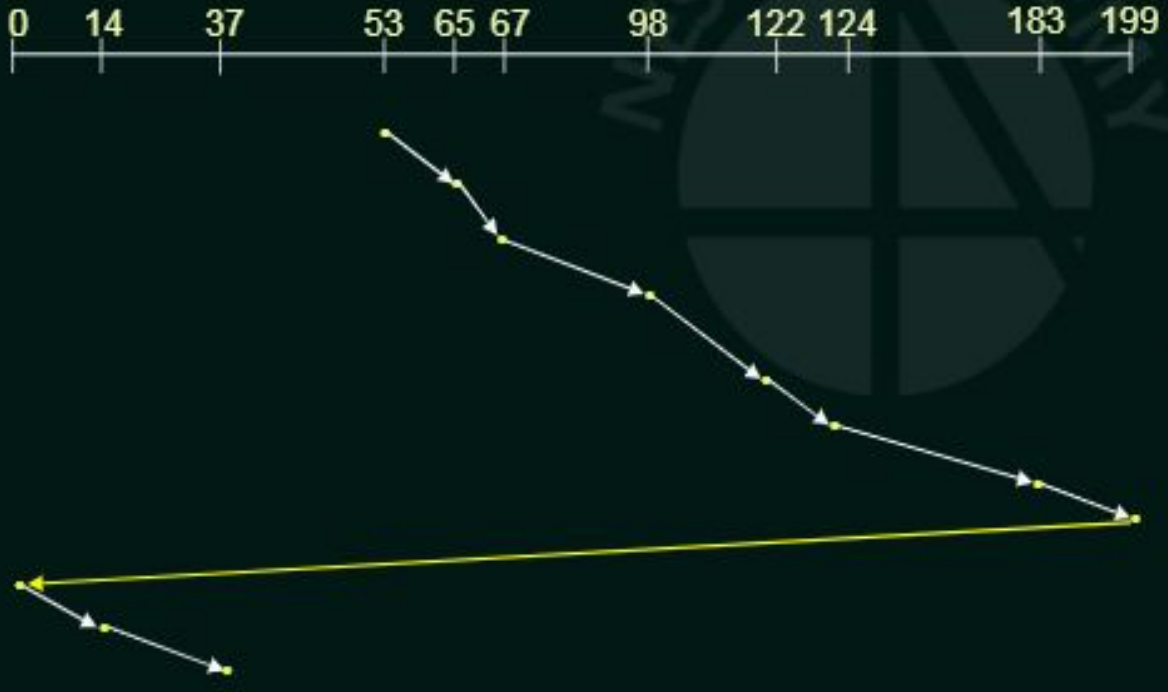
Advantages:

- This algorithm provides a more uniform wait time.

C-SCAN disk scheduling

Queue: 98, 183, 37,122, 14, 124, 65, 67

Head starts at: 53 (and direction of head movement is towards 199)



Disadvantages:

- Total head movement is more as compared to SCAN algorithm.

Disk Scheduling Algorithms

LOOK

- Both the SCAN and C-SCAN move the disk arm across the full width of the disk.
- But in LOOK algorithm, the arm goes only as far as the final request in each direction.
- Then, it reverses direction immediately, without going all the way to the end of the disk.

Example:

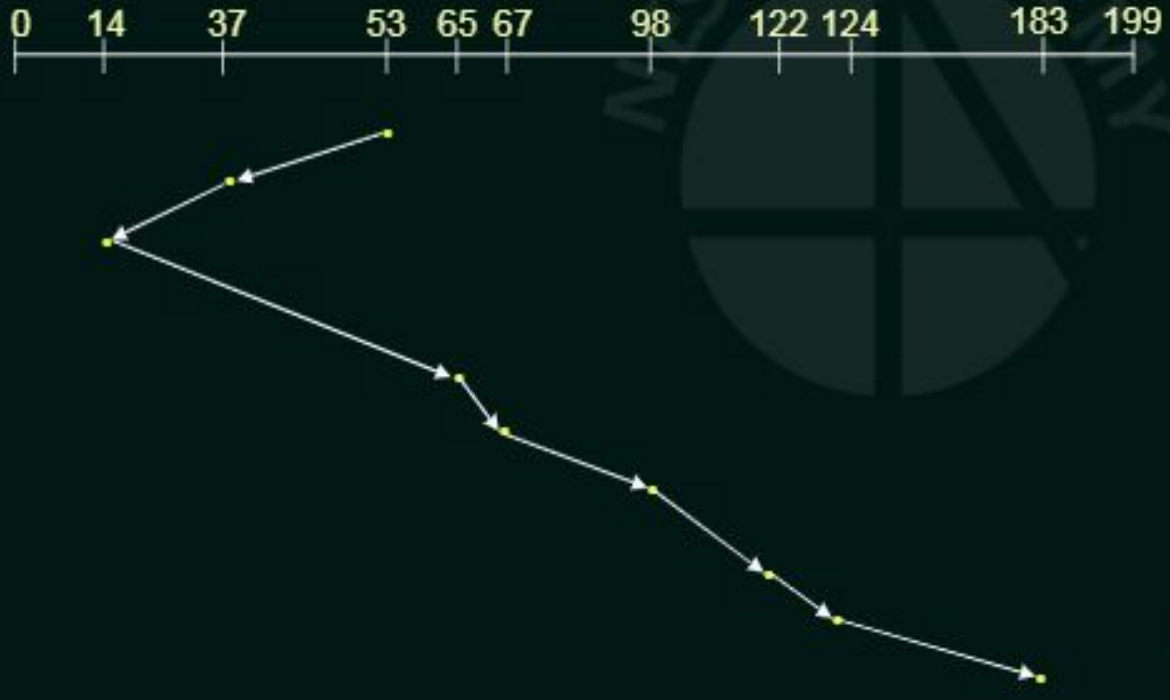
Consider a disk queue with requests for I/O to blocks on cylinders

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

LOOK disk scheduling

Queue: 98, 183, 37,122, 14, 124, 65, 67

Head starts at: 53 (and direction of head movement is towards 0)



Total head movement

=

(53-37) + (37-14) +
(65-14) + (67-65) +
(98-67) + (122-98) +
(124-122) + (183-124)

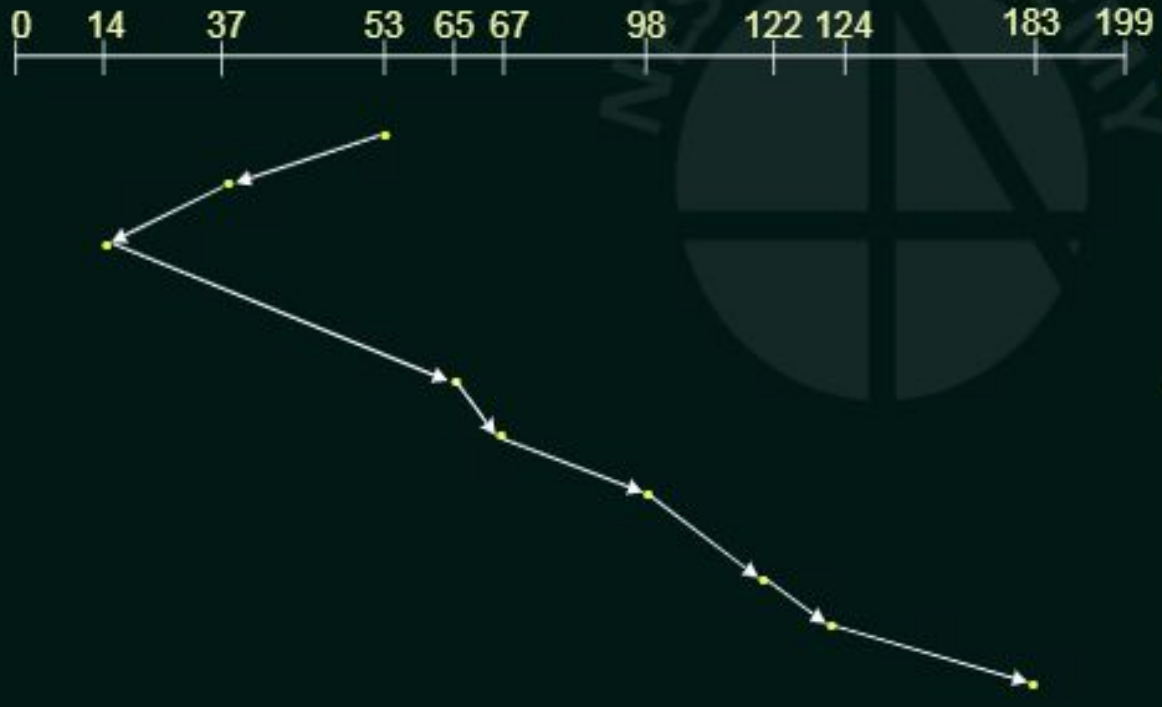
= 16 + 23 + 51 + 2 + 31 + 24 + 2 + 59

= 208

LOOK disk scheduling

Queue: 98, 183, 37,122, 14, 124, 65, 67

Head starts at: 53 (and direction of head movement is towards 0)



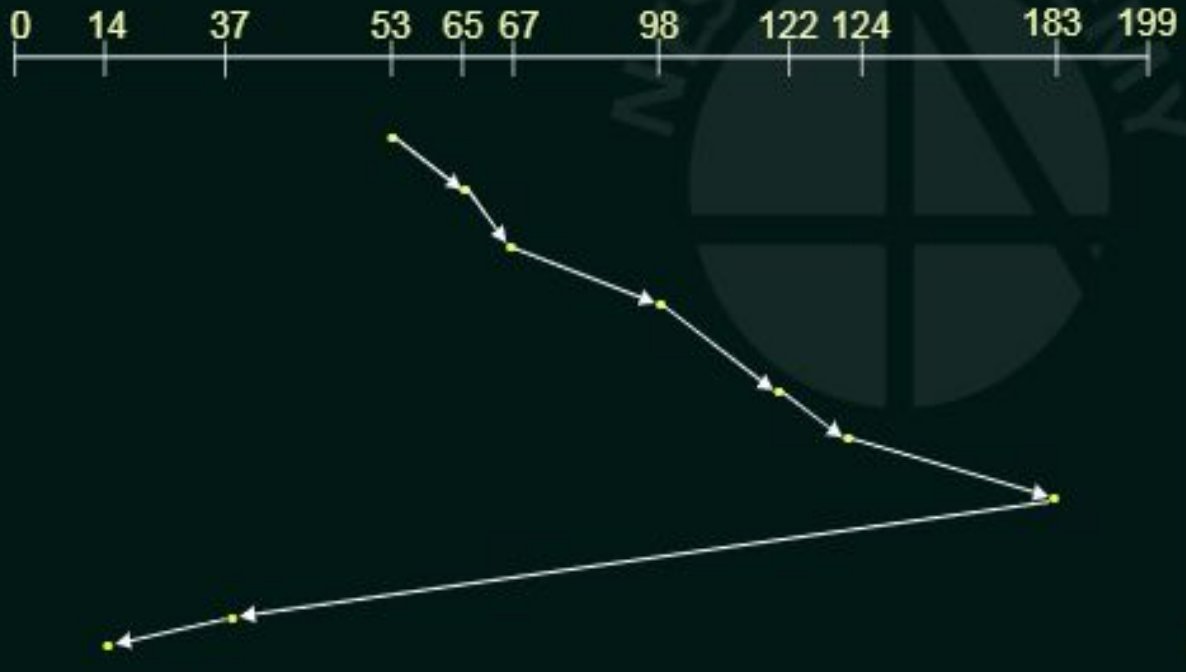
Advantages:

- The seek time is better (lesser) than all the algorithms discussed so far.

LOOK disk scheduling

Queue: 98, 183, 37,122, 14, 124, 65, 67

Head starts at: 53 (and direction of head movement is towards 199)



Total head movement

=

(65-53) + (67-65) +
(98-67) + (122-98) +
(124-122) + (183-124) +
(183-37) + (37-14)

= 12 + 2 + 31 + 24 + 2 + 59 +
146 + 23

= 299

Disk Scheduling Algorithms

Circular LOOK (C-LOOK)

- Like LOOK, C-LOOK algorithm moves the head in one direction servicing requests along the way.
- When the head reaches the last request, it immediately returns to the farthest request on the other end without servicing any requests on the return trip.
- Then from that position, it services the remaining request in the same direction like before.

Example:

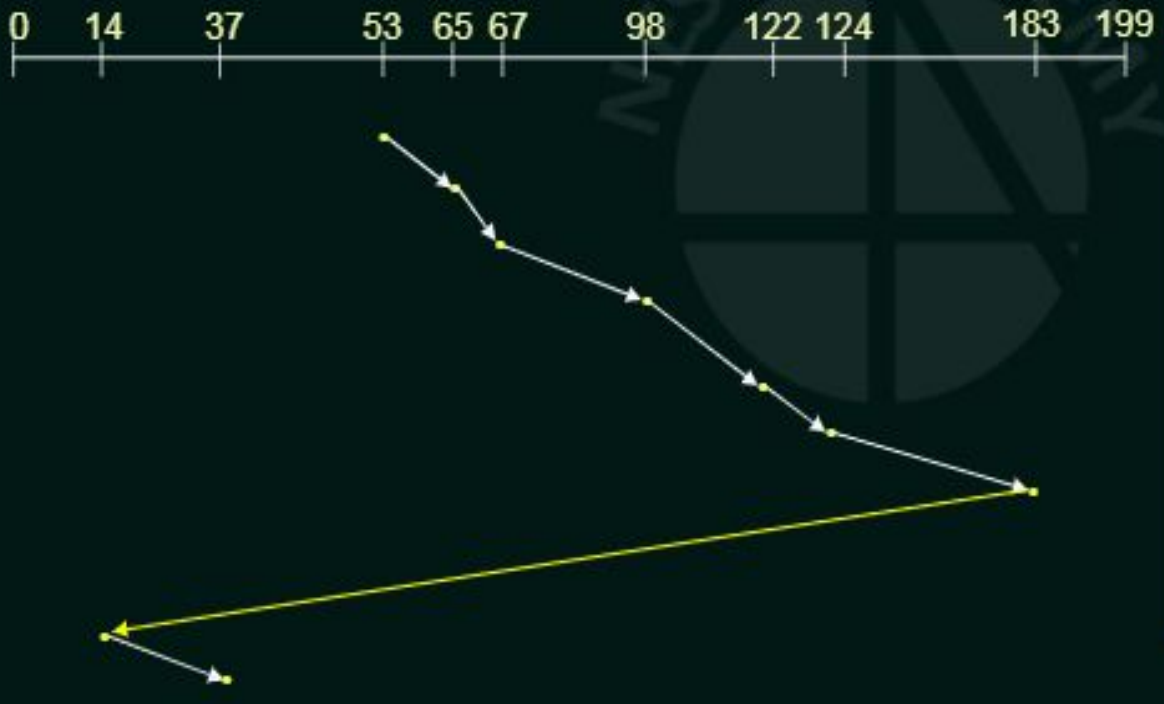
Consider a disk queue with requests for I/O to blocks on cylinders

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

C-LOOK disk scheduling

Queue: 98, 183, 37,122, 14, 124, 65, 67

Head starts at: 53 (and direction of head movement is towards 199)



Total head movement

=

(65-53) + (67-65) +
(98-67) + (122-98) +
(124-122) + (183-124) +
(183-14) + (37-14)

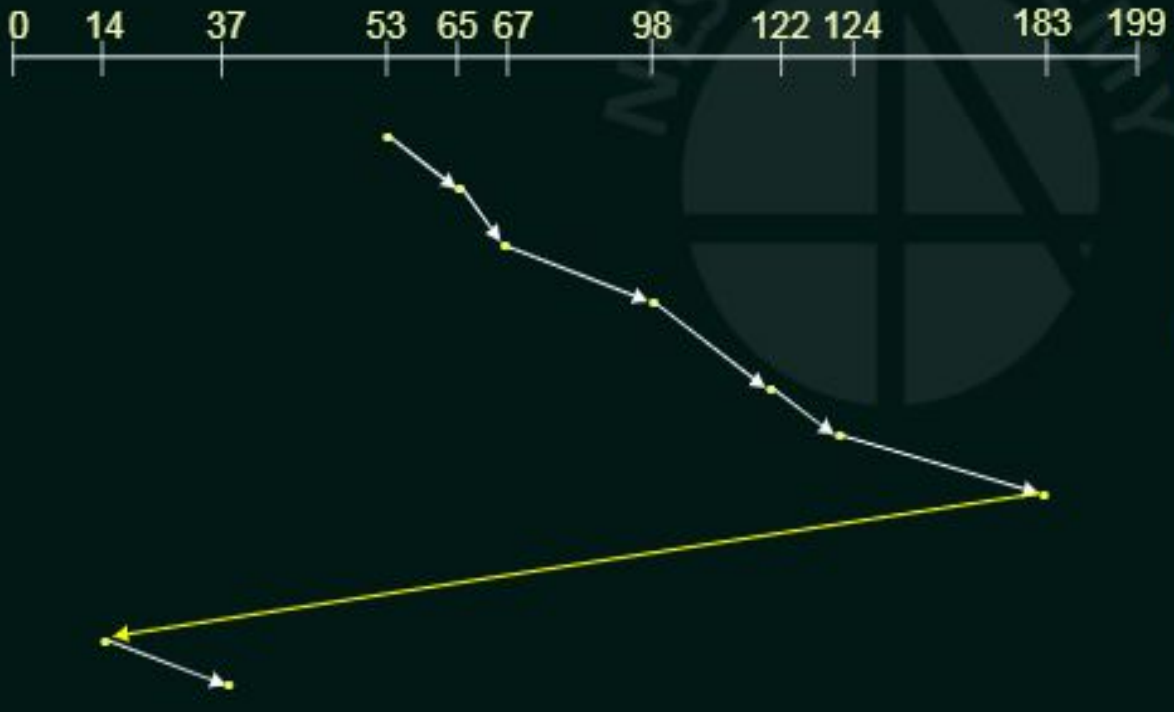
= 12 + 2 + 31 + 24 + 2 + 59 +
169 + 23

= 322

C-LOOK disk scheduling

Queue: 98, 183, 37,122, 14, 124, 65, 67

Head starts at: 53 (and direction of head movement is towards 199)



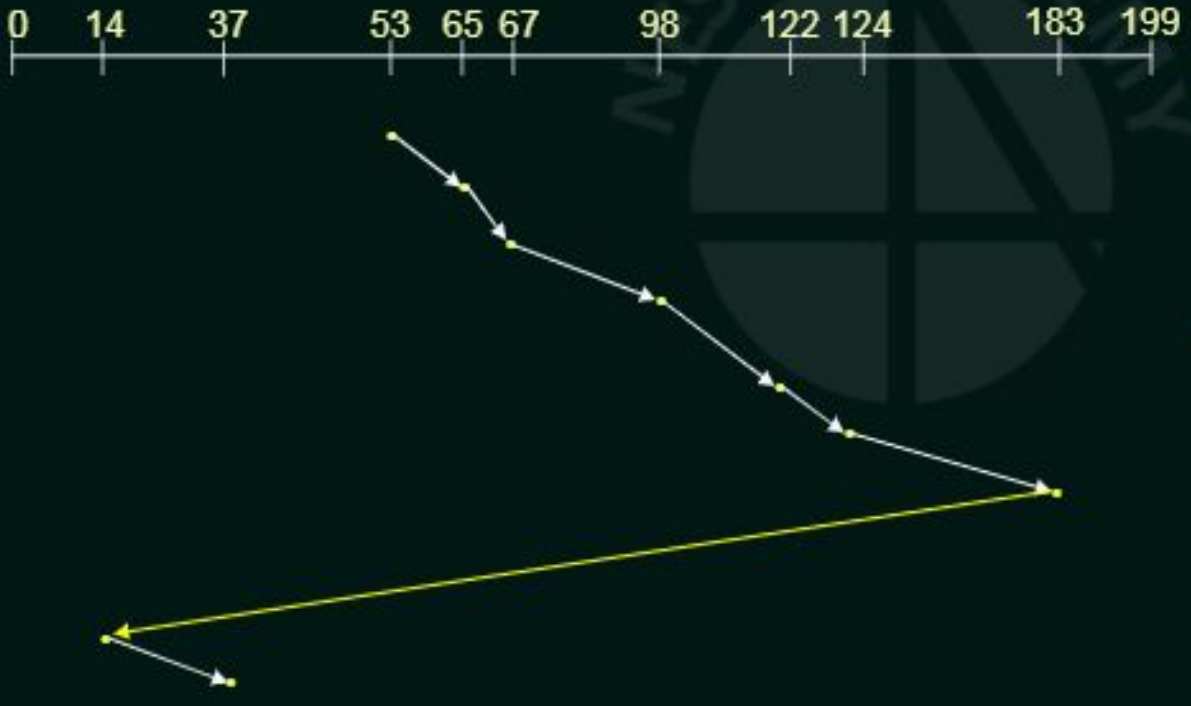
Advantages:

- This algorithm provides a more uniform wait time.

C-LOOK disk scheduling

Queue: 98, 183, 37,122, 14, 124, 65, 67

Head starts at: 53 (and direction of head movement is towards 199)



Disadvantages:

- Total head movement is more as compared to LOOK algorithm.

Selection of a Disk Scheduling Algorithm

We have studied about many disk-scheduling algorithms.

How do we choose the best one?

SSTF - Is common and has a natural appeal because it increases performance over FCFS.

SCAN & C-SCAN - Perform better for systems that place a heavy load on the disk, because they are less likely to cause a starvation problem.

For any particular list of requests, we can define an optimal order of retrieval, but the computation needed to find an optimal schedule may not justify the savings over SSTF or SCAN.

With any scheduling algorithm, performance depends heavily on the number and types of requests.

- Suppose that the queue usually has just one outstanding request.
- Then, all scheduling algorithms behave the same, because they have only one choice for where to move the disk head.
- They all behave like FCFS scheduling.

Requests for disk service can be greatly influenced by the file-allocation method.

- **Contiguous Allocation** – several requests that are close together on the disk → limited head movement.
- **Linked or Indexed Allocation** - blocks that are widely scattered on the disk → greater head movement.

The location of directories and index blocks is also important.

Every file must be opened to be used → opening a file requires searching the directory structure → the directories will be accessed frequently.

- Suppose that a directory entry is on the first cylinder and a file's data are on the final cylinder → the disk head has to move the entire width of the disk.
- If the directory entry were on the middle cylinder → the head would have to move, at most, one-half the width.
- Caching the directories and index blocks in main memory can also help to reduce the disk-arm movement, particularly for read requests.

Because of these complexities, the disk-scheduling algorithm should be written as a separate module of the operating system, so that it can be replaced with a different algorithm if necessary.

Either SSTF or LOOK is a reasonable choice for the default algorithm.

Disk Formatting

A new magnetic disk is a blank slate.

It is just a platter of a magnetic recording material.

Before a disk can store data, it must be divided into sectors that the disk controller can read and write.

This process is called low-level formatting, or physical formatting.

Low-level formatting

Low-level formatting fills the disk with a special data structure for each sector.

The data structure for a sector typically consists of:

- A header
- A data area (usually 512 bytes in size)
- A trailer

The header and trailer contain information used by the disk controller, such as a sector number and an error-correcting code (ECC).

Most hard disks are low-level-formatted at the factory as a part of the manufacturing process.

Logical formatting

To use a disk to hold files, the O.S. still needs to record its own data structures on the disk.

Step 1: Partition the disk into one or more groups of cylinders.
The operating system can treat each partition as though it were a separate disk.

Step 2: Logical formatting (or creation of a file system).
The O.S. stores the initial file-system data structures onto the disk. These data structures may include maps of free and allocated space (a FAT or modes) and an initial empty directory.

To increase efficiency, most file systems group blocks together into larger chunks, frequently called clusters.

Boot Block

For a computer to start running when it is powered up or rebooted it must have an initial program to run.

↓

The Bootstrap Program

↓

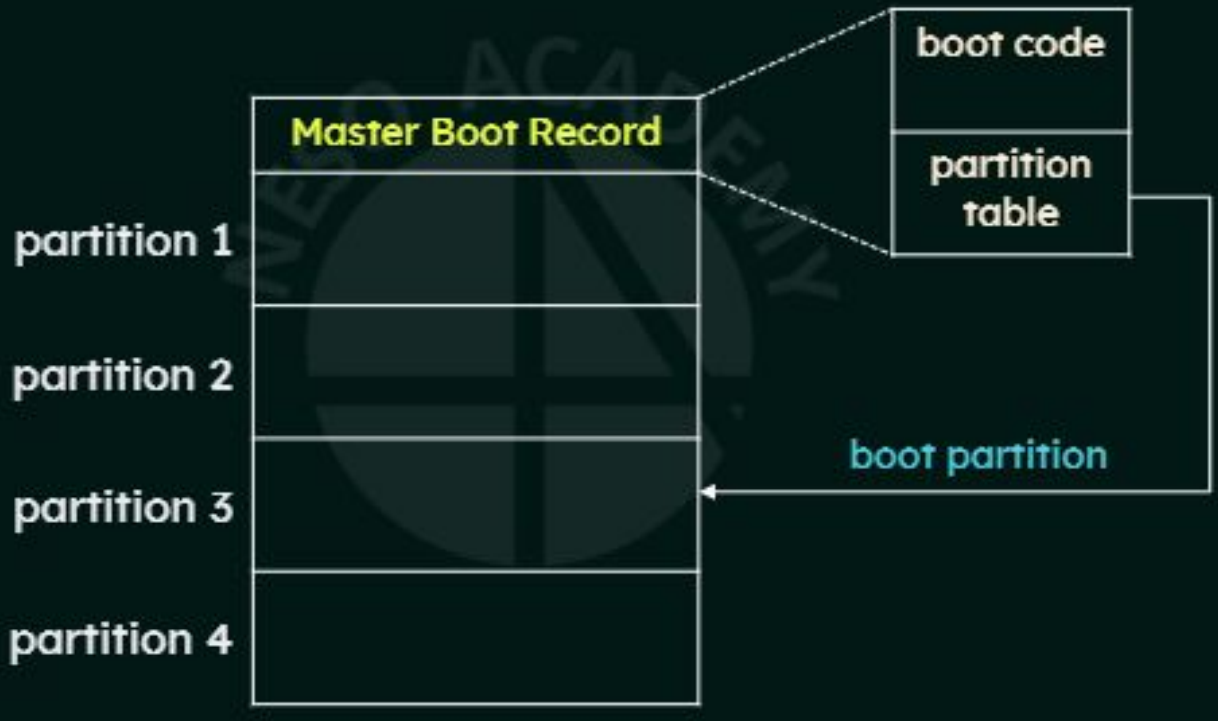
It initializes all aspects of the system, from CPU registers to device controllers and the contents of main memory, and then starts the operating system.

The bootstrap program finds the operating system kernel on disk, loads that kernel into memory, and jumps to an initial address to begin the operating-system execution.

Boot disk or System disk

- Most computers store the bootstrap in the read-only-memory (ROM).
- ROM needs no initialization and is at a fixed location that the processor can start executing when powered up or reset.
- ROM cannot be infected by a computer virus.
- **Problem:** Changing this bootstrap code requires changing the ROM, hardware chips.
- **To avoid this problem:** Most systems store a tiny **bootstrap loader program** in the boot ROM whose only job is to bring in a **full bootstrap program** from disk.
- The full bootstrap program can be changed easily: A new version is simply written onto the disk.
- The full bootstrap program is stored in "the boot blocks" at a fixed location on the disk.
- **A disk that has a boot partition is called a boot disk or system disk.**

Booting from disk in Windows 2000



Bad Blocks

Disks are prone to failure because they have moving parts and small tolerances.

Sometimes the failure is complete → the disk needs to be replaced and its contents restored from backup media to the new disk.

- More frequently, one or more sectors become defective.
- Most disks even come from the factory with bad blocks.
- Depending on the disk and controller in use, these blocks are handled in a variety of ways.

Handling bad blocks

- Manual handling - used in simple disks
- Sector sparing or forwarding
- Sector slipping

- used in more sophisticated disks

Handling bad blocks

Manual handling:

- The MS-DOS format command performs logical formatting and, as a part of the process, scans the disk to find bad blocks.
- If format finds a bad block, it writes a special value into the corresponding FAT entry to tell the allocation routines not to use that block.

- If blocks go bad during normal operation, a special program (such as chkdsk) must be run manually to search for the bad blocks and to lock them away as before.
- Data that resided on the bad blocks usually are lost.

Handling bad blocks

Sector sparing or forwarding:

- The controller maintains a list of bad blocks on the disk.
- The list is initialized during the low-level formatting at the factory and is updated over the life of the disk.
- Low-level formatting also sets aside spare sectors not visible to the operating system.
- The controller can be told to replace each bad sector logically with one of the spare sectors.

Handling bad blocks

Example of Sector sparing or forwarding:

- The operating system tries to read logical block 87.
- The controller calculates the ECC and finds that the sector is bad. It reports this finding to the operating system.
- The next time the system is rebooted, a special command is run to tell the SCSI controller to replace the bad sector with a spare.
- After that, whenever the system requests logical block 87, the request is translated into the replacement sector's address by the controller.

Handling bad blocks

Sector slipping:

- Suppose that logical block 17 becomes defective and the first available spare follows sector 202.
- Then, sector slipping remaps all the sectors from 17 to 202, moving them all down one spot.
- That is, sector 202 is copied into the spare, then sector 201 into 202, and then 200 into 201, and so on, until sector 18 is copied into sector 19.
- Slipping the sectors in this way frees up the space of sector 18, so sector 17 can be mapped to it.
- Sequential order is maintained.

Swap-Space Use

Swap space is used in various ways by different operating systems, depending on the memory-management algorithms in use.

Example:

1. Systems that implement swapping may use swap space to hold an entire process image, including the code and data segments.
2. Paging systems may simply store pages that have been pushed out of main memory.

The amount of swap space needed on a system can depend on:

- The amount of physical memory.
- The amount of virtual memory it is backing.
- The way in which the virtual memory is used.

It can range from a few megabytes of disk space to gigabytes.

Estimating the amount of swap space required in a system

It may be safer to overestimate than to underestimate the amount of swap space required.

- **Underestimating** - If a system runs out of swap space it may be forced to abort processes or may crash entirely.
- **Overestimating** - Wastes disk space that could otherwise be used for files, but it does no other harm.

Depends from system to system:

Solaris - Suggests setting swap space equal to the amount by which virtual memory exceeds pageable physical memory.

Linux – Suggests setting swap space to double the amount of physical memory, although most Linux systems now use considerably less swap space.

Swap-Space Location

A swap space can reside in one of two places:

1. It can be carved out of the normal file system.
2. It can be in a separate disk partition.

1. It can be carved out of the normal file system.

If the swap space is simply a large file within the file system, normal file-system routines can be used to create it, name it, and allocate its space.

Advantages	Disadvantages
1. Easy to implement.	1. It is inefficient. 2. Navigating the directory structure and the disk-allocation data structures takes time and (potentially) extra disk accesses. 3. External fragmentation can greatly increase swapping times by forcing multiple seeks during reading or writing of a process image.

2. It can be in a separate disk partition.

Swap space can be created in a separate raw partition, as no file system or directory structure is placed in this space.

- A separate swap-space storage manager is used to allocate and deallocate the blocks from the raw partition.
- This manager uses algorithms optimized for speed rather than for storage efficiency, because swap space is accessed much more frequently than file systems (when it is used).

- Internal fragmentation may increase, but this trade-off is acceptable because the life of data in the swap space generally is much shorter than that of files in the file system.
- Swap space is reinitialized at boot time so any fragmentation is short-lived.
- Adding more swap space requires repartitioning the disk.

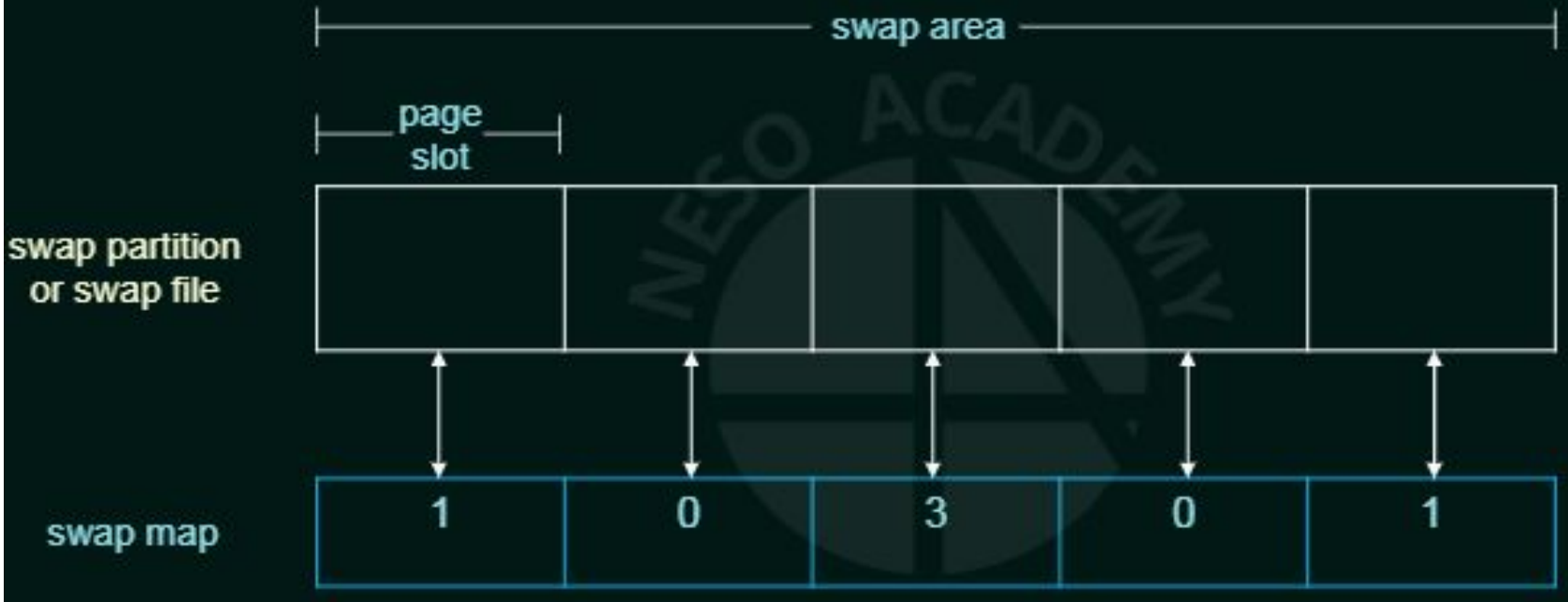
An Example of Swap-Space Management

Swap-space management in Linux

- Swap space is only used for anonymous memory or for regions of memory shared by several processes.
- Linux allows one or more swap areas to be established.
- A swap area may be in either:

  A swap file on a regular file system or a raw swap partition.
- Each swap area consists of a series of 4-KB page slots, which are used to hold swapped pages.
- Associated with each swap area is a swap map—an array of integer counters, each corresponding to a page slot in the swap area.

The data structures for swapping on Linux systems



Swap map

An array of integer counters, each corresponding to a page slot in the swap area.

- If the value of a counter is 0, the corresponding page slot is available.
- Values greater than 0 indicate that the page slot is occupied by a swapped page.
- The value of the counter indicates the number of mappings to the swapped page.
- Example:

  A value of 3 indicates that the swapped page is mapped to three different processes (which can occur if the swapped page is storing a region of memory shared by three processes).

RAID Structure

Redundant arrays of inexpensive disks

- Disk drives have continued to get smaller and cheaper, so it is now economically feasible to attach many disks to a computer system.
- RAID is a way of storing the same data in different places on multiple hard disks or SSDs to protect data in the case of failure.
- It improves the rate at which data can be read or written, if the disks are operated in parallel.
- It improves the reliability of data storage, because redundant information can be stored on multiple disks. Thus, failure of one disk does not lead to loss of data.
- In the past, RAID was used as a cost effective alternative to large expensive disks.
- Today, they are used for their higher reliability and higher data-transfer rate, rather than for economic reasons.

RAID Structure

Redundant arrays of independent disks

- Disk drives have continued to get smaller and cheaper, so it is now economically feasible to attach many disks to a computer system.
- RAID is a way of storing the same data in different places on multiple hard disks or SSDs to protect data in the case of failure.
- It improves the rate at which data can be read or written, if the disks are operated in parallel.
- It improves the reliability of data storage, because redundant information can be stored on multiple disks. Thus, failure of one disk does not lead to loss of data.
- In the past, RAID was used as a cost effective alternative to large expensive disks.
- Today, they are used for their higher reliability and higher data-transfer rate, rather than for economic reasons.

Working of RAID

- It places data on multiple disks and allows I/O operations to overlap in a balanced way, thus improving performance.
- As the use of multiple disks increases the mean time between failures, storing data redundantly also increases fault tolerance.
- To the OS, RAID arrays appear as a single logical drive.
- RAID employs the techniques of disk mirroring or disk striping.

Copies identical data onto more than one drive.

Spread data over multiple disk drives.

Improvement of Reliability via Redundancy

Reliability:

The chance that some disk out of a set of N disks will fail is much higher than the chance that a specific single disk will fail.

E.g.

Suppose that the mean time to failure of a single disk = 100,000 hours.

The mean time to failure of some disk in an array of 100 disks = 100,000/100

= 1000 hours = 41.66 days.

If we store only one copy of the data, then each disk failure will result in loss of a significant amount of data—and such a high rate of data loss is unacceptable.

Solution: Introduce redundancy.

Thus, even if a disk fails, data are not lost.

Mirroring

Mirroring is the technique of duplicating all the disks.

- It is simple but expensive.
- A logical disk then consists of two physical disks, and every write is carried out on both disks.
- If one of the disks fails, the data can be read from the other.
- Data will be lost only if the second disk fails before the first failed disk is replaced.
- The mean time to failure—where failure is the loss of data—of a mirrored volume (made up of two disks, mirrored) depends on two factors:
    1) Mean time to failure of the individual disks.
    2) The mean time to repair.

Mirroring

Suppose that the failures of the two disks are independent:

- If the mean time to failure of a single disk = 100,000 hours
- If the mean time to repair = 10 hours
- The mean time to data loss of a mirrored disk system = $100,000^2/(2 * 10)$

  $= 500 * 10^6$ hours

  = 57,000 years

Improvement in Performance via Parallelism

Performance:

With multiple disks, we can improve the transfer rate by striping data across the disks.

Data Striping:

In its simplest form, data striping consists of splitting the bits of each byte across multiple disks → **bit-level striping.**

Example: **A byte - 11001110**

Example: **A byte - 11001110**

- We write bit i of each byte to disk i.
- The array of eight disks can be treated as a single disk with sectors that are eight times the normal size and have eight times the access rate.
- Every disk participates in every access (read or write).
- The number of accesses that can be processed per second is about the same as on a single disk, but each access can read eight times as many data in the same time as on a single disk.

Bit-level striping can be generalized to include a number of disks that either is a multiple of 8 or divides 8.

Example:

If we use an array of four disks bits i and 4 + i of each byte go to disk i.

Example: A byte - 11001110

Block-level striping:

- Blocks of a file are striped across multiple disks → with n disks, block i of a file goes to disk (i mod n) + 1.
- Other levels of striping, such as bytes of a sector or sectors of a block, also are possible.

Parallelism in a disk system, as achieved through striping, has two main goals:

1. Increase the throughput of multiple small accesses (that is, page accesses) by load balancing.

2. Reduce the response time of large accesses.

RAID Levels (Part-1)

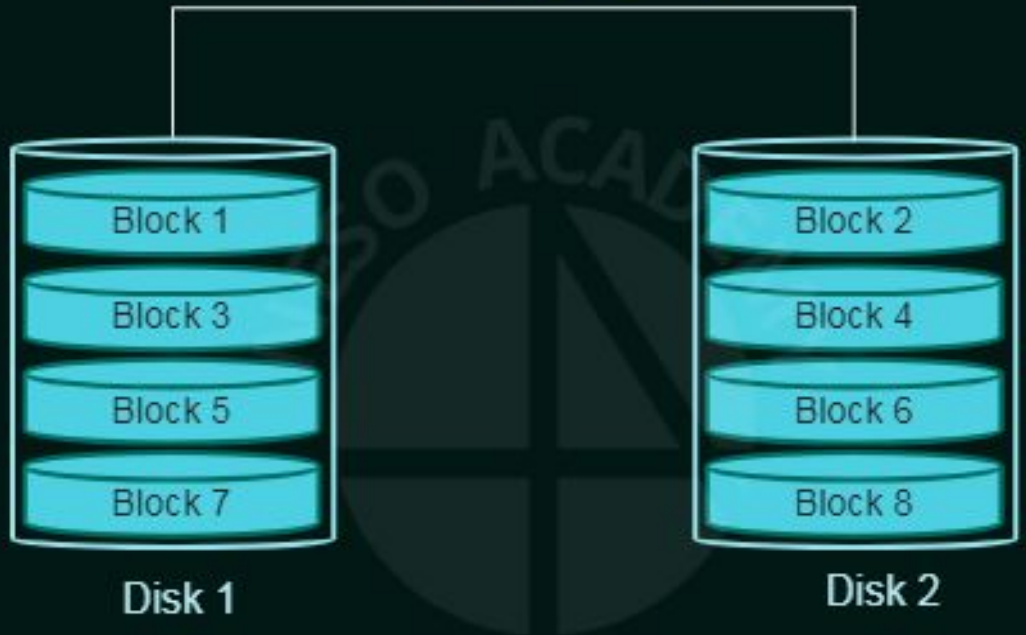
Mirroring → Provides high reliability, but it is expensive.

Striping → Provides high data-transfer rates, but it does not improve reliability.

- Numerous schemes to provide redundancy at lower cost by using the idea of disk striping combined with "parity" have been proposed.

- These schemes have different cost-performance trade-offs and are classified according to levels called **RAID levels**.

RAID Level 0

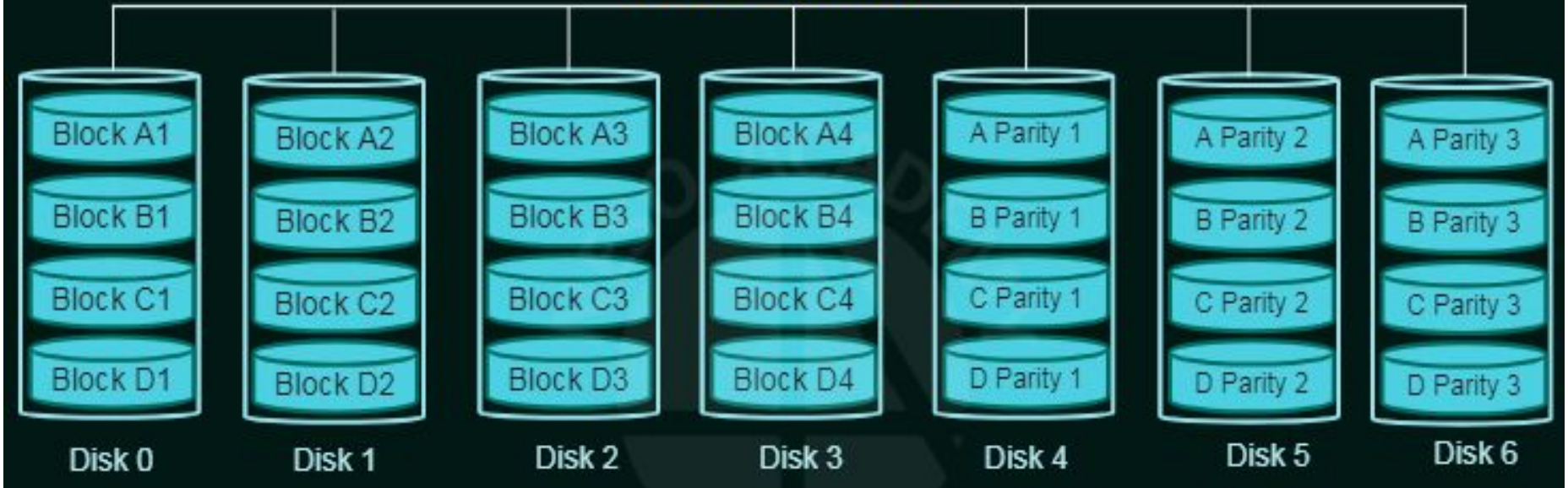


- RAID level 0 refers to disk arrays with striping at the level of blocks but without any redundancy (such as mirroring or parity bits).
- It offers the best performance, but it does not provide fault tolerance.

RAID Level 1

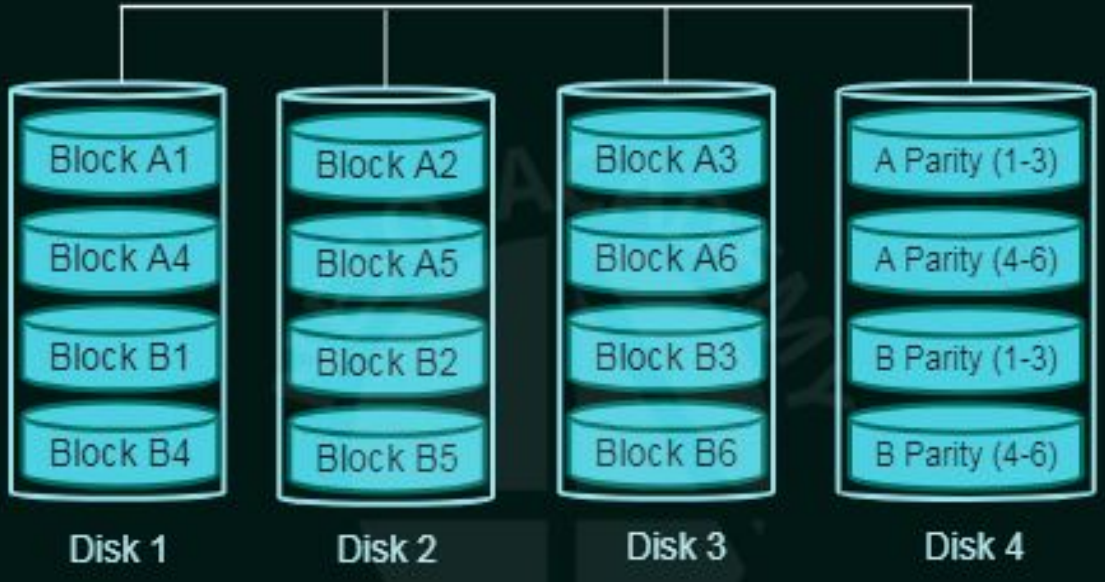
- RAID level 1 refers to disk mirroring.
- Read performance is improved since either disk can be read at the same time.
- Write performance is the same as for single disk storage.

RAID Level 2



- RAID level 2 is also known as memory-style error-correcting code (ECC) organization.
- This configuration uses striping across disks, with some disks storing ECC information.
- If one of the disks fails, the remaining bits of the byte and the associated error-correction bits can be read from other disks and used to reconstruct the damaged data.

RAID Level 3



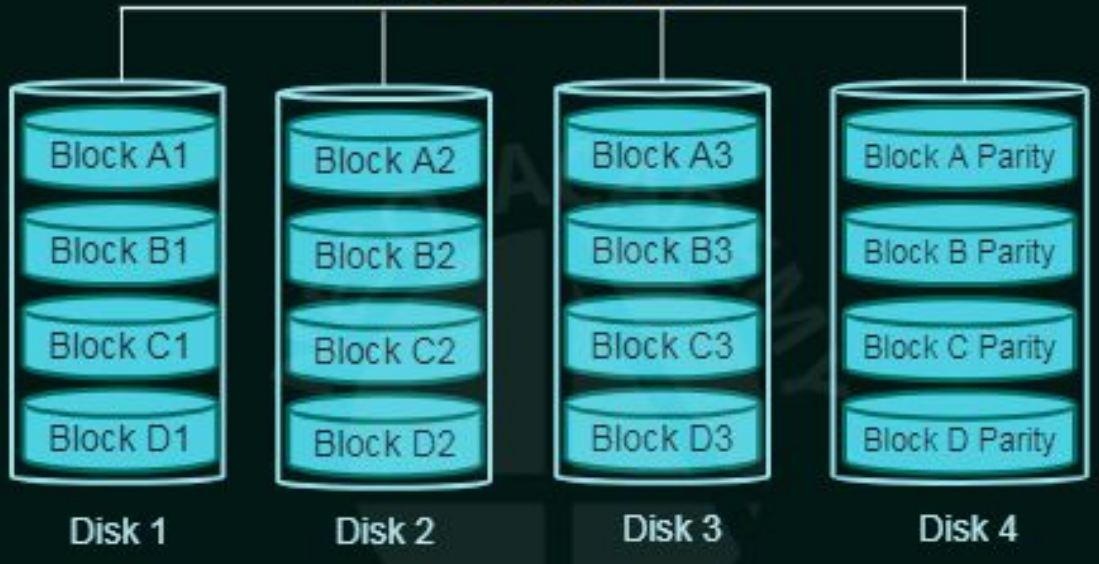
- RAID level 3, or bit-interleaved parity organization, takes into account the fact that, unlike memory systems, disk controllers can detect whether a sector has been read correctly, so a single parity bit can be used for error correction as well as for detection.
- It uses striping and dedicates one drive to storing parity information.

RAID Levels (Part-2)

- RAID Level – 4
- RAID Level – 5
- RAID Level – 6
- RAID Level – 0 + 1
- RAID Level – 1 + 0

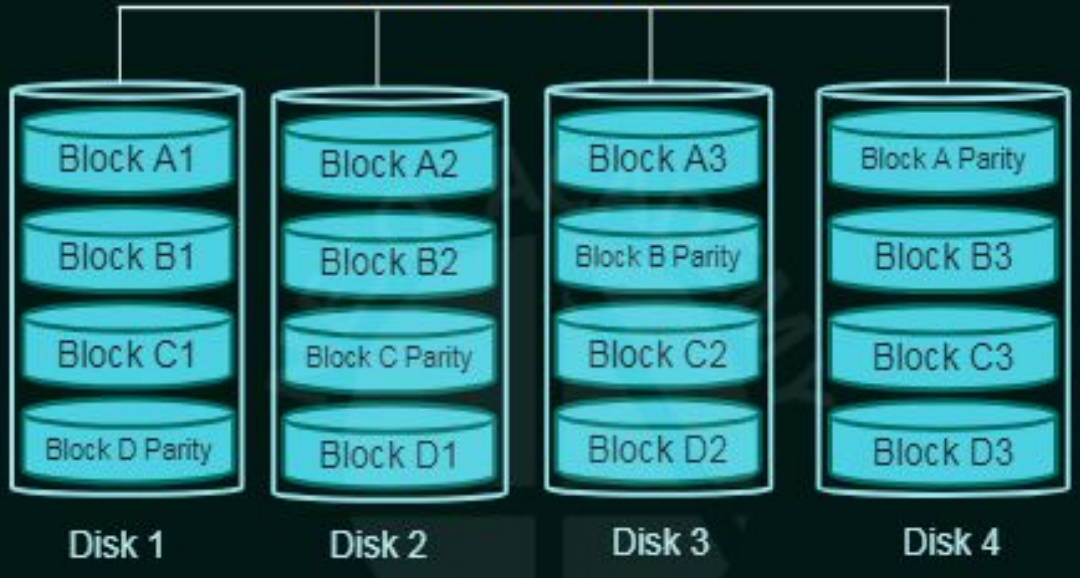
Nested RAID Levels

RAID Level 4



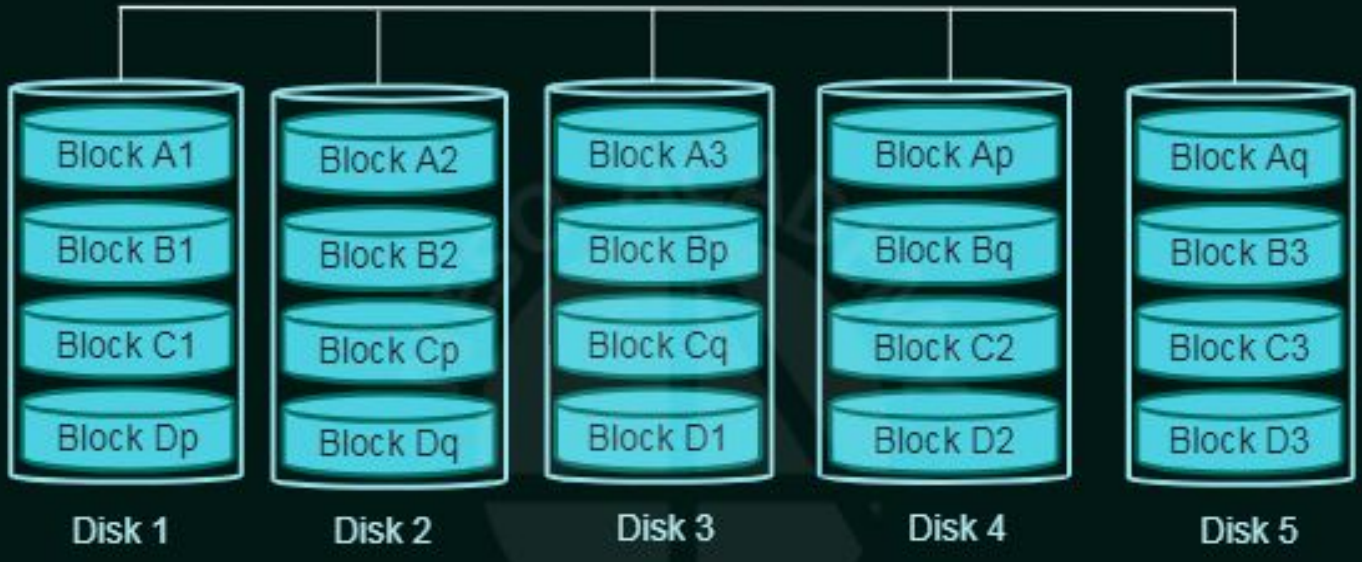
- RAID level 4, or block-interleaved parity organization, uses block-level striping.
- In addition it keeps a parity block on a separate disk for corresponding blocks from N other disks.
- If one of the disks fails, the parity block can be used with the corresponding blocks from the other disks to restore the blocks of the failed disk.

RAID Level 5



- RAID level 5, or block-interleaved distributed parity, spreads data and parity among all N + 1 disks, rather than storing data in N disks and parity in one disk.
- For each block, one of the disks stores the parity, and the others store data.
- By spreading the parity across all the disks in the set, RAID 5 avoids the potential overuse of a single parity disk that can occur with RAID 4.

RAID Level 6



- RAID level 6, also called the P + Q redundancy scheme, is much like RAID level 5 but stores extra redundant information to guard against multiple disk failures.
- The use of additional parity enables the array to continue to function even if two disks fail simultaneously.
- RAID 6 arrays often have slower write performance than RAID 5 arrays.

Nested RAID Levels

RAID Level 0 + 1

- RAID level 0 + 1 refers to a combination of RAID levels 0 and 1.
- RAID 0 provides the performance, while RAID 1 provides the reliability.
- Generally, this level provides better performance than RAID 5.
- It is common in environments where both performance and reliability are important.
- But, it doubles the number of disks needed for storage.

RAID Level 1 + 0

- Disks are mirrored in pairs, and then the resulting mirror pairs are striped.
- Advantage over Level 0 +1:

  If a single disk fails in RAID 0 + 1, the entire stripe is inaccessible, leaving only the other stripe available. With a failure in RAID 1+0, the single disk is unavailable, but its mirrored pair is still available, as are all the rest of the disks.

Stable-Storage Implementation

Information residing in stable storage is never lost.

How to implement such storage?

- We need to replicate the needed information on multiple storage devices (usually disks) with independent failure modes.

- We need to coordinate the writing of updates in a way that guarantees that:
  - A failure during an update will not leave all the copies in a damaged state.
  - When we are recovering from a failure, we can force all copies to a consistent and correct value, even if another failure occurs during the recovery.

A disk write results in one of three outcomes:

1. **Successful completion:**

   The data were written correctly on disk.

2. **Partial failure:**

   A failure occurred in the midst of transfer, so only some of the sectors were written with the new data, and the sector being written during the failure may have been corrupted.

3. **Total failure:**

   The failure occurred before the disk write started, so the previous data values on the disk remain intact.

Whenever a failure occurs during writing of a block, the system needs to detect it and invoke a recovery procedure to restore the block to a consistent state.

To do that, the system must maintain two physical blocks for each logical block.

An output operation is executed as follows:

1. Write the information onto the first physical block.
2. When the first write completes successfully, write the same information onto the second physical block.
3. Declare the operation complete only after the second write completes successfully.

During recovery from a failure, each pair of physical blocks is examined.

- If both are the same and no detectable error exists, then no further action is necessary.
- If one block contains a detectable error, then we replace its contents with the value of the other block.
- If neither block contains a detectable error, but the blocks differ in content, then we replace the content of the first block with that of the second.
- This recovery procedure ensures that a write to stable storage either succeeds completely or results in no change.

Tertiary-Storage

The main objective of the tertiary storage level is to provide huge storage capacity at low cost.

Because cost is so important, in practice, tertiary storage is built with removable media.

The most common examples are:

- Floppy disks
- Tapes
- Read-only, write-once, and rewritable CDs and DVDs.

Removable Disks

Removable disks are one kind of tertiary storage.

Examples:

- **Floppy disks** - Made from a thin, flexible disk coated with magnetic material and enclosed in a protective plastic case. Usual size it can hold ≈ 1 MB.
- **Magneto-optic disk** - It records data on a rigid platter coated with magnetic material, but the recording technology is quite different from that for a magnetic disk.
- **Optical disk** - Optical disks do not use magnetism at all. Instead, they use special materials that can be altered by laser light to have relatively dark or bright spots.
  E.g. Phase-change disk
- **WORM (write-once, read-many-times) disks** – Consists of a thin aluminum film sandwiched between two glass or plastic platters. To write a bit, the drive uses a laser light to burn a small hole through the aluminum. This burning cannot be reversed.

Mass-Storage Structure

Solved Problem - 1

UGC NET CS 2016 JULY-II

If the Disk head is located initially at track 32, find the number of disk moves required with FCFS scheduling criteria if the disk queue of I/O blocks requests are

98, 37, 14, 124, 65, 67.

(A) 320

(B) 322

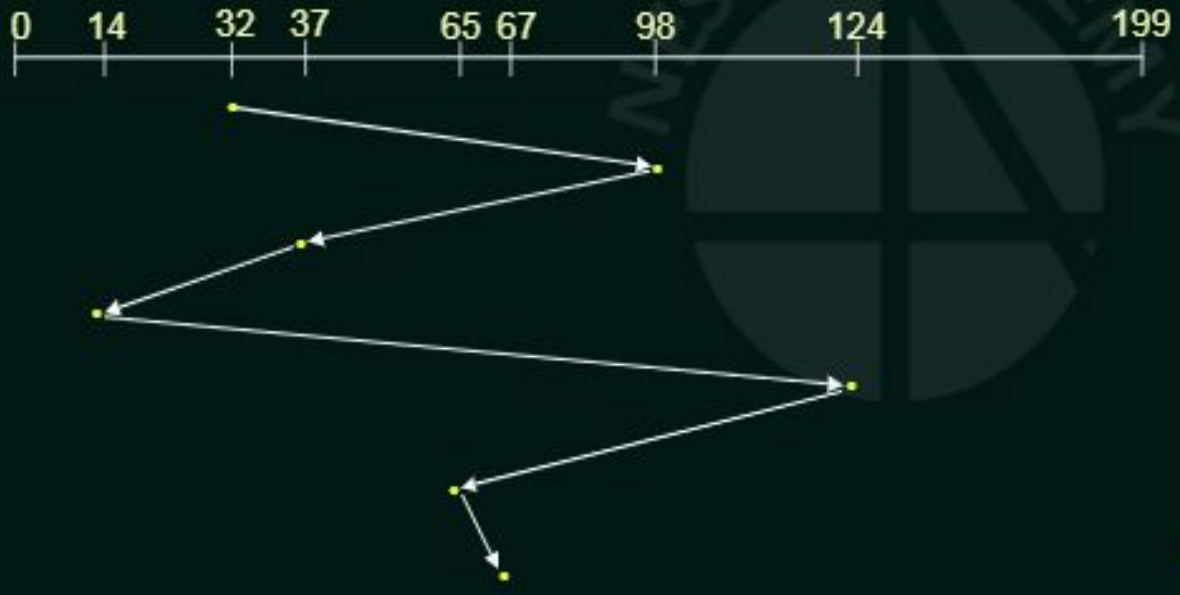
(C) 321

(D) 319

FCFS disk scheduling

Queue: 98, 37, 14, 124, 65, 67

Head starts at: 32



Total head movement

=

(32-98) + (98-37) +
(37-14) + (124-14) +
(124-65) + (67-65)

= 66 + 61 + 23 + 110 + 59 + 2

= 321

Mass-Storage Structure

Solved Problem - 1

UGC NET CS 2016 JULY-II

If the Disk head is located initially at track 32, find the number of disk moves required with FCFS scheduling criteria if the disk queue of I/O blocks requests are

98, 37, 14, 124, 65, 67.

(A) 320

(B) 322

(C) 321

(D) 319

Mass-Storage Structure

Solved Problem - 2

ISRO CS 2014

There are 200 tracks on a disc platter and the pending requests have come in the order

36, 69, 167, 76, 42, 51, 126, 12 and 199.

Assume the arm is located at the 100 track and moving towards track 199.

If sequence of disc access is

126, 167, 199, 12, 36, 42, 51, 69 and 76

then which disc access scheduling policy is used?

(A) Elevator

(B) Shortest Seek-Time First

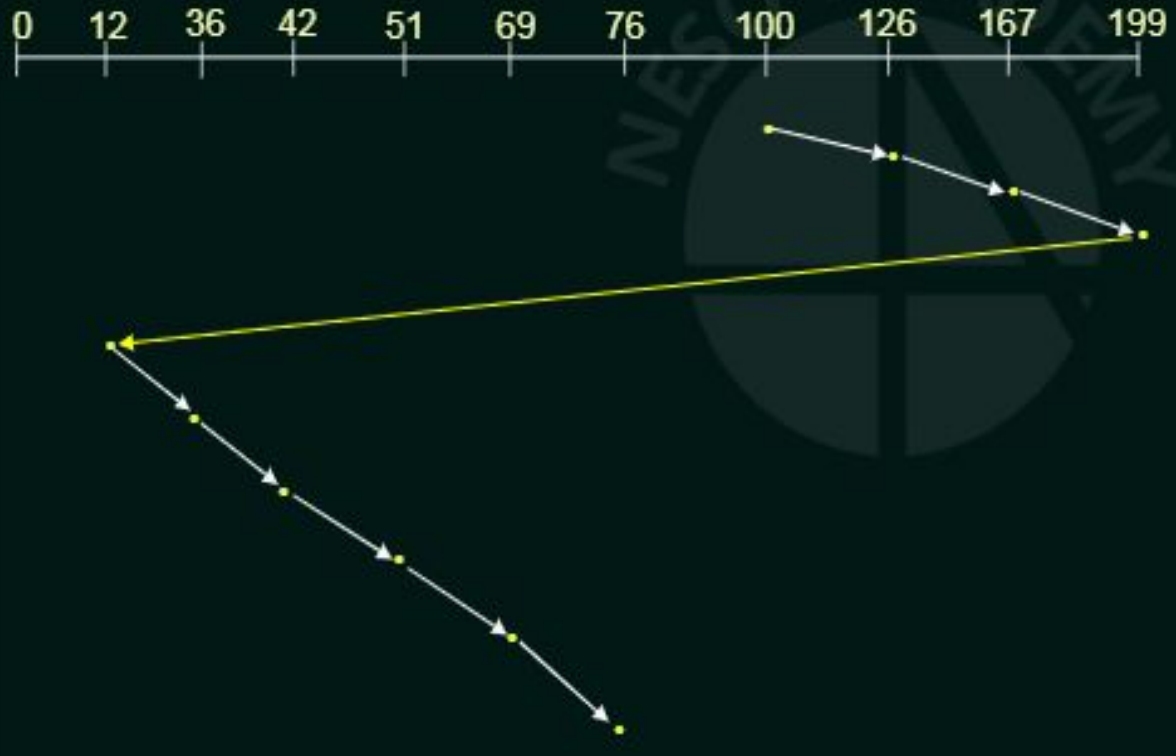
(C) C-SCAN

(D) First Come First Served

Queue: 36, 69, 167, 76, 42, 51, 126, 12, 199

Sequence of Access: 126, 167, 199, 12, 36, 42, 51, 69, 76

Head starts at: 100



Options:

(A) Elevator

(B) Shortest Seek-Time First

(C) C-SCAN

(D) First Come First Served

Mass-Storage Structure

Solved Problem - 2

ISRO CS 2014

There are 200 tracks on a disc platter and the pending requests have come in the order

36, 69, 167, 76, 42, 51, 126, 12 and 199.

Assume the arm is located at the 100 track and moving towards track 199.

If sequence of disc access is

126, 167, 199, 12, 36, 42, 51, 69 and 76

then which disc access scheduling policy is used?

(A) Elevator

(B) Shortest Seek-Time First

(C) C-SCAN

(D) First Come First Served

Mass-Storage Structure

Solved Problem - 3

GATE 2018

Consider a storage disk with 4 platters (numbered as 0, 1, 2 and 3), 200 cylinders (numbered as 0, 1, … , 199), and 256 sectors per track (numbered as 0, 1, … 255). The following 6 disk requests of the form [sector number, cylinder number, platter number] are received by the disk controller at the same time:

[120, 72, 2], [180, 134, 1], [60, 20, 0], [212, 86, 3], [56, 116, 2], [118, 16, 1]

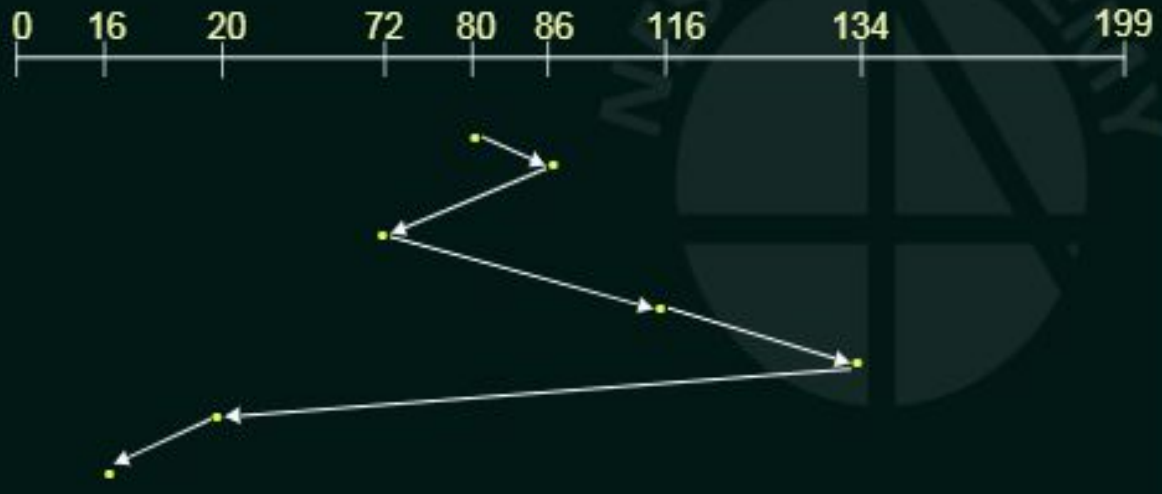
Currently head is positioned at sector number 100 of cylinder 80, and is moving towards higher cylinder numbers. The average power dissipation in moving the head over 100 cylinders is 20 milliwatts and for reversing the direction of the head movement once is 15 milliwatts. Power dissipation associated with rotational latency and switching of head between different platters is negligible.

The total power consumption in milliwatts to satisfy all of the above disk requests using the Shortest Seek Time First disk scheduling algorithm is _____ .

SSTF disk scheduling

Queue: 72, 134, 20, 86, 116, 16

Head starts at: 80



Total head movement

=

(86-80) + (86-72) +
(116-72) + (134-116) +
(134-20) + (20-16)

= 6 + 14 + 44 + 18 + 114 + 4

= 200

Total head movement = 200

Given:

- Average power dissipation in moving the head over 100 cylinders = 20 mW
- For reversing the direction of the head movement once = 15 mW

Power dissipated by 1 head movement = 20/100 = 0.2 mW

Power dissipated by 200 movements = 0.2 x 200 = 40 mW

Number of times change in head direction occurs = 3

Power dissipated in reversing head direction = 3 x 15 = 45 mW

Total power consumption = (40 + 45) mW = 85 mW

Mass-Storage Structure

Solved Problem - 3

GATE 2018

Consider a storage disk with 4 platters (numbered as 0, 1, 2 and 3), 200 cylinders (numbered as 0, 1, … , 199), and 256 sectors per track (numbered as 0, 1, … 255). The following 6 disk requests of the form [sector number, cylinder number, platter number] are received by the disk controller at the same time:

[120, 72, 2], [180, 134, 1], [60, 20, 0], [212, 86, 3], [56, 116, 2], [118, 16, 1]

Currently head is positioned at sector number 100 of cylinder 80, and is moving towards higher cylinder numbers. The average power dissipation in moving the head over 100 cylinders is 20 milliwatts and for reversing the direction of the head movement once is 15 milliwatts. Power dissipation associated with rotational latency and switching of head between different platters is negligible.

The total power consumption in milliwatts to satisfy all of the above disk requests using the Shortest Seek Time First disk scheduling algorithm is __85 mW__ .

Mass-Storage Structure

Solved Problem - 4

GATE 2016

Consider a disk queue with requests for I/O to blocks on cylinders

47, 38, 121, 191, 87, 11, 92, 10.

The C-LOOK scheduling algorithm is used.

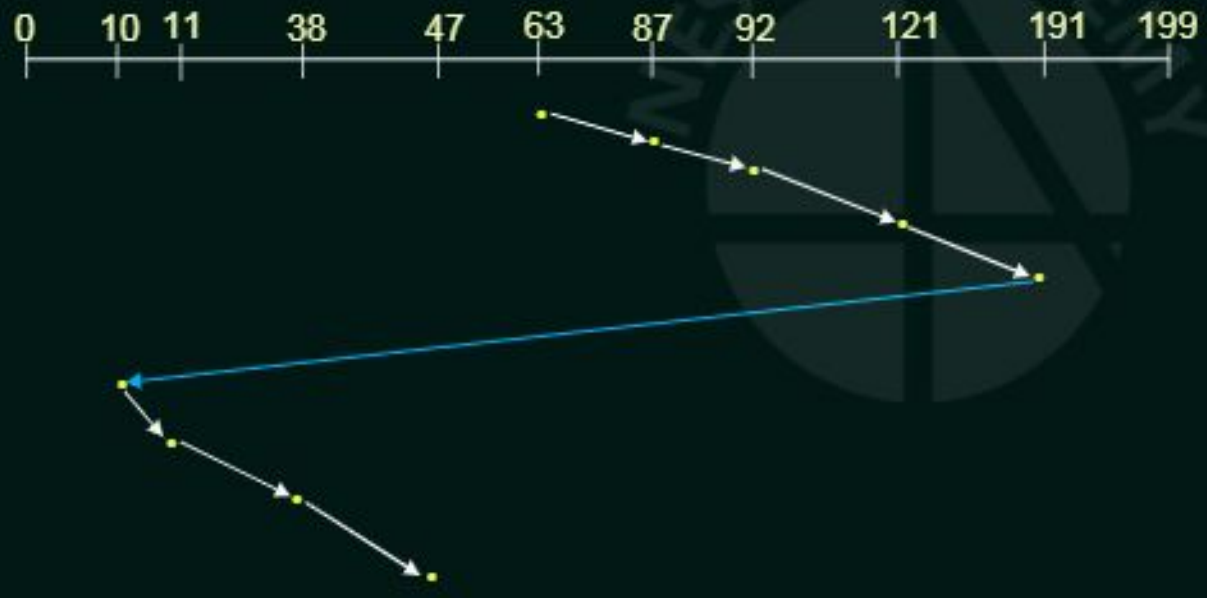
The head is initially at cylinder number 63, moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199.

The total head movement (in number of cylinders) incurred while servicing these requests is _____ .

C-LOOK disk scheduling

Queue: 47, 38, 121, 191, 87, 11, 92, 10

Head starts at: 63



Total head movement

=

(87-63) + (92-87) +
(121-92) + (191-121) +
(191-10) + (11-10) +
(38-11) + (47-38)

= 24 + 5 + 29 + 70 + 181 + 1
+ 27 + 9

= 346

Mass-Storage Structure

Solved Problem - 4

GATE 2016

Consider a disk queue with requests for I/O to blocks on cylinders

47, 38, 121, 191, 87, 11, 92, 10.

The C-LOOK scheduling algorithm is used.

The head is initially at cylinder number 63, moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199.

The total head movement (in number of cylinders) incurred while servicing these requests is 346 .

Mass-Storage Structure

Solved Problem - 5

GATE 2020

Consider the following five disk access requests of the form

(request id, cylinder number)

that are present in the disk scheduler queue at a given time.

(P, 155), (Q, 85), (R, 110), (S, 30), (T, 115). Assume the head is positioned at cylinder 100.

The scheduler follows Shortest Seek Time First scheduling to service the requests.

Which one of the following statements is FALSE ?

(A) T is serviced before P

(B) Q is serviced after S, but before T

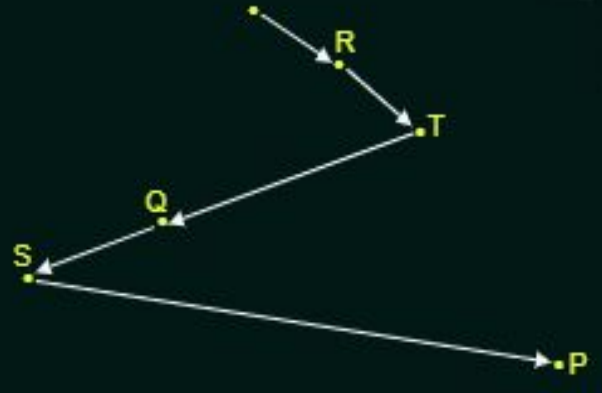
(C) The head reverses its direction of movement between servicing of Q and P

(D) R is serviced before P

SSTF disk scheduling

Queue: (P, 155), (Q, 85), (R, 110), (S, 30), (T, 115)

Head starts at: 100



Options:

(A) T is serviced before P

(B) Q is serviced after S, but before T

(C) The head reverses its direction of movement between servicing of Q and P

(D) R is serviced before P

Mass-Storage Structure

Solved Problem - 5

GATE 2020

Consider the following five disk access requests of the form

(request id, cylinder number)

that are present in the disk scheduler queue at a given time.

(P, 155), (Q, 85), (R, 110), (S, 30), (T, 115). Assume the head is positioned at cylinder 100.

The scheduler follows Shortest Seek Time First scheduling to service the requests.

Which one of the following statements is FALSE ?

(A) T is serviced before P

(B) Q is serviced after S, but before T

(C) The head reverses its direction of movement between servicing of Q and P

(D) R is serviced before P

Mass-Storage Structure

Solved Problem - 6

Suppose the following disk request sequence (track numbers) for a disk with 100 tracks is given:

45, 20, 90, 10, 50, 60, 80, 25, 70.

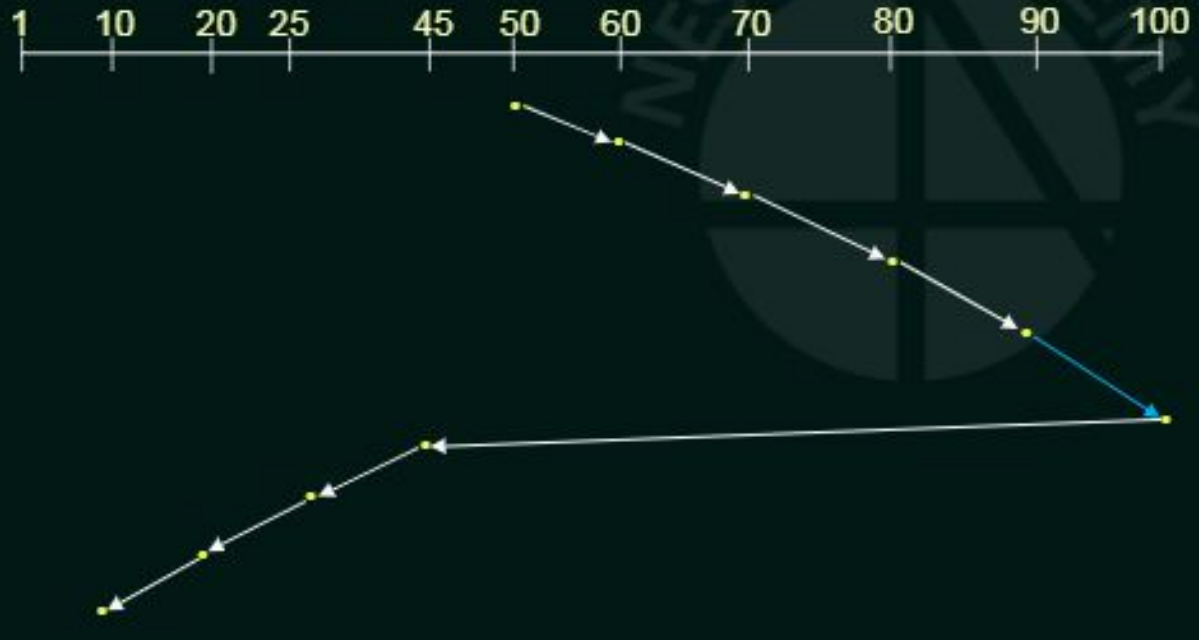
Assume that the initial position of the R/W head is on track 50.

The additional distance that will be traversed by the R/W head when the SCAN (Elevator) algorithm (assuming that SCAN algorithm moves towards 100 when it starts execution) is used compared to the Shortest Seek Time First (SSTF) algorithm is _____ tracks.

SCAN (Elevator) disk scheduling

Queue: 45, 20, 90, 10, 50, 60, 80, 25, 70

Head starts at: 50



Total head movement

=

(60-50) + (70-60) +
(80-70) + (90-80) +
(100-90) + (100-45) +
(45-25) + (25-20) +
(20-10)

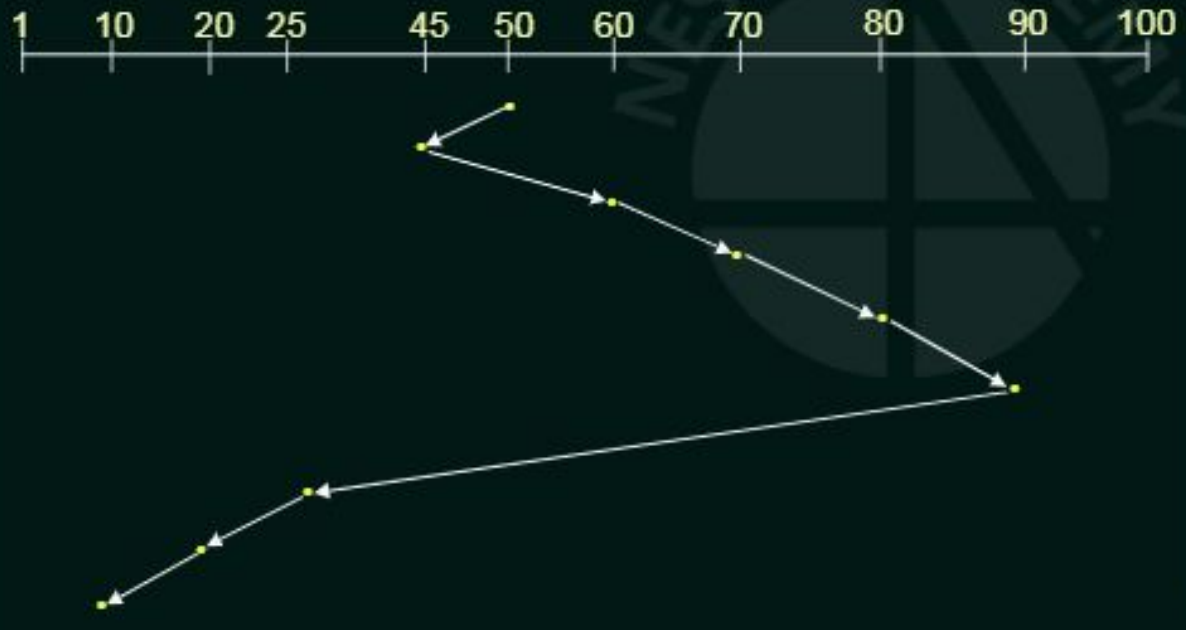
= 10 + 10 + 10 + 10 + 10 + 55
+ 20 + 5 + 10

= 140

SSTF disk scheduling

Queue: 45, 20, 90, 10, 50, 60, 80, 25, 70

Head starts at: 50



Total head movement
=
(50-45) + (60-45) +
(70-60) + (80-70) +
(90-80) + (90-25) +
(25-20) + (20-10)

= 5 + 15 + 10 + 10 + 10 + 65
+ 5 + 10

= 130

Queue: 45, 20, 90, 10, 50, 60, 80, 25, 70

Head starts at: 50

Total head movement (SCAN)

=

(60-50) + (70-60) +
(80-70) + (90-80) +
(100-90) + (100-45) +
(45-25) + (25-20) +
(20-10)

= 10 + 10 + 10 + 10 + 10 + 55
+ 20 + 5 + 10

= 140

Total head movement (SSTF)

=

(50-45) + (60-45) +
(70-60) + (80-70) +
(90-80) + (90-25) +
(25-20) + (20-10)

= 5 + 15 + 10 + 10 + 10 + 65
+ 5 + 10

= 130

Additional distance traversed by the R/W head when the SCAN algorithm is used compared to the SSTF

= 140 - 130
= 10

Mass-Storage Structure

Solved Problem - 6

Suppose the following disk request sequence (track numbers) for a disk with 100 tracks is given:

45, 20, 90, 10, 50, 60, 80, 25, 70.

Assume that the initial position of the R/W head is on track 50.

The additional distance that will be traversed by the R/W head when the SCAN (Elevator) algorithm (assuming that SCAN algorithm moves towards 100 when it starts execution) is used compared to the Shortest Seek Time First (SSTF) algorithm is

___10___ tracks.