T. Y. B. Sc.
COMPUTER SCIENCE
SEMESTER-VI

NEW SYLLABUS
CBCS PATTERN

OPERATING SYSTEMS-II

Dr. Ms. MANISHA BHARAMBE
Mrs. VEENA GANDHI

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

Contents ...

1. Process Deadlocks 1.1 – 1.56

2. File System Management 2.1 – 2.36

3. Disk Scheduling 3.1 – 3.22

4. Introduction to Distributed Operating Systems and Architecture 4.1 – 4.64

5. Mobile Operating Systems 5.1 – 5.44

■■■

CHAPTER 3

Disk Scheduling

Objectives...

- To understand Concept of Disk
- To learn Disk Scheduling
- To study different Disk Scheduling Algorithms

3.0 INTRODUCTION

- File systems must be accessed in an efficient manner, especially with hard drives, which are the slowest part of a computer.
- As a computer deals with multiple processes over a period of time, a list of requests to access the disk builds up. For efficiency purposes, all requests (from all processes) are aggregated together.
- The technique that the operating system uses to determine which requests to satisfy first is called disk scheduling
- Disk scheduling is a technique used by the operating system to schedule multiple requests for accessing the disk. The algorithms used for disk scheduling are called as disk scheduling algorithms.
- In most system there are many processes that are running simultaneously. Often, many processes request I/O operations from / to the hard drive.
- The algorithm used to select which request is going to be satisfied first is called a disk scheduling algorithm.
- Various disk scheduling algorithms are used in accessing data and transferring data. These are disk scheduling algorithms are FCFS, SSTF, SCAN and LOOK.

3.1 OVERVIEW

- As we know, a process needs two types of time, CPU time and IO time. For I/O, it requests the operating system to access the disk.
- However, the operating system must be fair enough to satisfy each request and at the same time, the operating system must maintain the efficiency and speed of process execution.

- The technique that the operating system uses to determine the request which is to be satisfied next is called disk scheduling.
- The main purpose of the disk scheduling algorithm is to select a disk request from the queue of IO requests and decide the schedule when this request will be processed.
- Hard disk is magnetic storage medium for a computer. Hard disks are flat circular plates made of aluminum or glass and coated with a magnetic material.
- A floppy disk is a flexible disk with a magnetic coating on it. It is packaged inside a protective plastic envelope.
- The floppy disks are one of the oldest types of portable storage devices that could store up to 1.44 MB of data but now they are not used due to very less memory storage.

3.1.1 Disk Structure

- In modern computers, most of the secondary storage is in the form of magnetic disks. Hence, knowing the structure of a magnetic disk is necessary to understand how the data in the disk is accessed by the computer.

Structure of a Magnetic Disk:

- A magnetic disk contains several platters. Each platter is divided into circular shaped tracks. The length of the tracks near the centre is less than the length of the tracks farther from the centre.
- Each track is further divided into sectors, as shown in the figure. 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.
- Tracks of the same distance from the centre form a cylinder. A read-write head is used to read data from a sector of the magnetic disk.

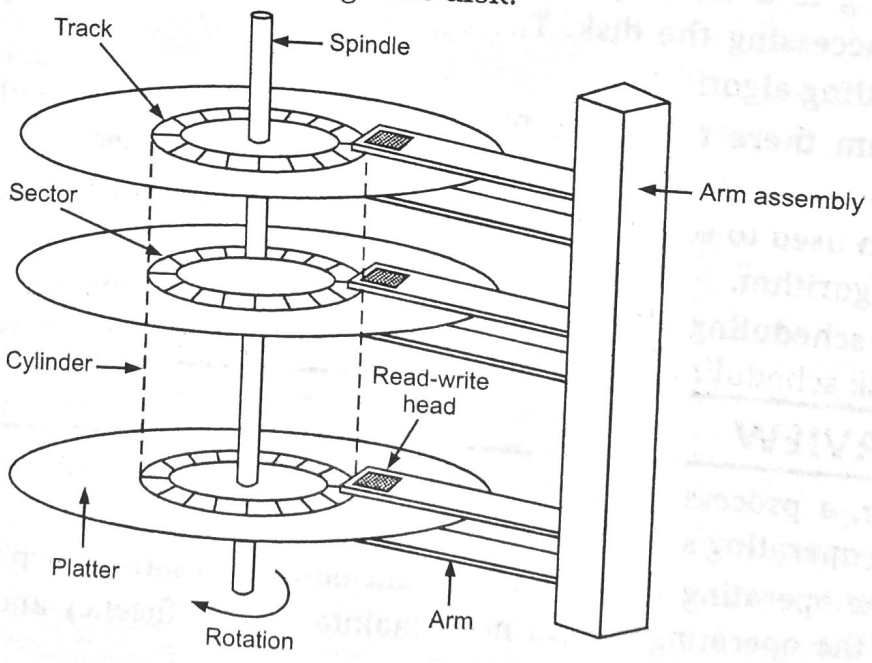

Fig. 3.1: Disk Structure

- The speed of the disk is measured as two parts:
  - **Transfer Rate** is the rate at which the data moves from disk to the computer.
  - **Random Access Time** is the sum of the seek time and rotational latency.
  - The **seek time** is the time for the disk arm to move the head to the required cylinder containing the desired track.
  - The **rotational latency** is the additional time for the disk to rotate the desired sector to the disk head.
  - The **disk bandwidth** is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.
  - A disk drive is attached to a computer by a set of wires called an I/O bus. Several kinds of buses are available, including Advanced Technology Attachment (ATA), Serial ATA (SATA), eSATA, Universal Serial Bus (USB), and Fiber Channel (FC).
  - The data transfers on a bus are carried out by special electronic processors called controllers. The **host controller** is the controller at the computer end of the bus.
  - A **disk controller** is built into each disk drive. To perform a disk I/O operation, the computer places a command into the host controller, typically using memory-mapped I/O ports.
- Even though the disk is arranged as sectors and tracks physically, the data is logically arranged and addressed as an array of blocks of fixed size. The size of a block can be 512 or 1024 bytes.
- The logical block is the smallest unit of transfer. The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially. In this way, each sector in the disk will have a logical address. Sector 0 is the first sector of the first track on the outermost cylinder.
- Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.

3.2 DISK SCHEDULING

- On a typical multiprogramming system, there will usually be multiple disk access requests at any point of time. So those requests must be scheduled to achieve good efficiency.
- The OS is responsible for minimizing access time and maximizing bandwidth for disks. Access time has two major components – Seek time and Rotational latency.
- Whenever a process needs I/O to or from the disk, it issues a system call to the operating system.
- The request specifies several pieces of information like whether operation is input or output, disk address, memory address for the transfer, number of sectors to be transferred etc.

- Requests are serviced immediately if the disk is not busy, if disk is busy then requests are queued and disk scheduling is used to choose the next request to service.

3.2.1 Disk Performance Parameters

- When a disk drive is operating, the disk is rotating at constant speed. To read or write, the head must be positioned at the desired track and at the beginning of the desired sector on the track.
- Track selection involves moving the head in a movable-head system or electronically selecting one head on a fixed head system.
- On a movable-head system, the time taken to position the head at the track is known as seek time.
- Once the track is selected, the disk controller waits until the appropriate sector rotates to line up with the head.
- The time it takes to reach the beginning of the desired sector is known as rotational delay or rotational latency.
- The sum of the seek time, (for movable head system) and the rotational delay is termed as access time of the disk, the time it takes to get into appropriate position (track and sector) to read or write.
- Once the head is in position, the read or write operation is then performed as the sector moves under the head, and the data transfer takes place.

Performance parameters are as follows:

1. **Seek Time:**

- Seek time is the time required to move the disk arm to the required track. The seek time is approximated as,

$$T_s = m \times n + s$$

where, T_s = estimated seek time
n = number of tracks traversed
m = constant that depends on the disk drive
s = startup time

2. **Rotational Delay:**

- Disk drive generally rotates at 3600 rpm, i.e. to make one revolution it takes around 16.7 ms. Thus on the average, the rotational delay will be 8.3 ms.

3. **Transfer Time:**

- The transfer time to or from the disk depends on the rotational speed of the disk and it is estimated as,

$$T = \frac{b}{rN}$$

where,

T = Transfer time

b = Number of bytes to be transferred

N = Numbers of bytes on a track

r = Rotational speed, in revolution per second.

4. **Access Time:**

Access Time = Seek Time + Rotational Delay + Transfer Rate

Thus, the total average access time can be expressed as,

$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

where T_s is the average seek time.

5. **Disk Response Time:**

- Response Time is the average of time spent by a request waiting to perform its I/O operation. Average Response time is the response time of all requests.
- Variance Response Time is a measure of how individual requests are serviced with respect to average response time.

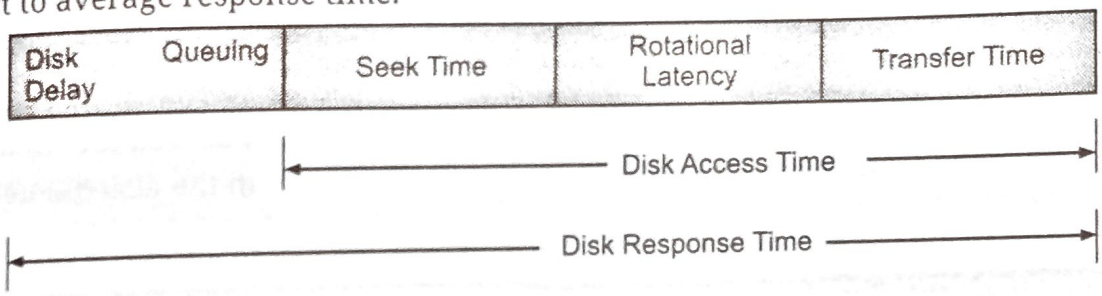

Fig. 3.2: Disk Performance Parameter

- Disks are a potential bottleneck for system performances and storage system reliability.
- The disk access time is relatively higher than the time required to access data from main memory and perform CPU operation.
- Also the disk drive contains some mechanical parts and it involves mechanical movement, so the failure rate is also high.
- The disk performance has been improving continuously; microprocessor performance has improved much more rapidly.
- A disk array is an arrangement of several disks, organized to increase performance and improve reliability of the resulting storage system. Performance is increased through data striping. Reliability is improved through redundancy.
- Disk arrays that implement a combination of data striping and redundancy are called Redundant Arrays of Independent Disks (RAID).

3.2.2 Scheduling Algorithms

- Disk scheduling is done by operating systems to schedule I/O requests arriving for disk.
- Disk scheduling is important because:
  1. Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
  2. Two or more requests may be far from each other so can result in greater disk arm movement.
  3. Hard drives are one of the slowest parts of a computer system and thus need to be accessed in an efficient manner.
- Disk scheduling is similar to process scheduling. The disk scheduling algorithm that gives minimum average seek time, minimum rotational latency and minimum variance response time is better.
- Some of the disk scheduling algorithms are described in following sections.

3.2.2.1 FCFS

- The simplest form of disk scheduling is the First-Come, First-Served (FCFS) algorithm. In FCFS, the requests are addressed in the order they arrive in the disk queue.
- The FCFS algorithm is intrinsically fair to all processes, but it generally does not provide the fastest service.
- For example, consider, for example, a disk queue with requests for I/O to block on cylinders 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the cylinder 50 and the tail cylinder being at 199.
- All incoming requests are placed at the end of the queue. If the disk head is initially at cylinder 50, it will first move from 50 to 95, 95 then to 180, 34, 119, 11, 123, 62 and finally to 64.

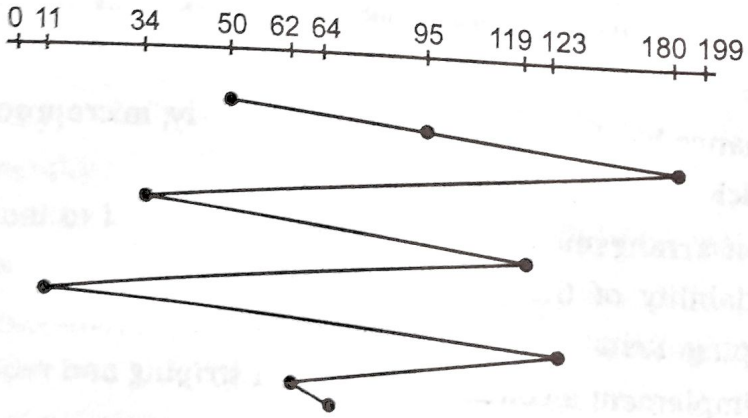

Fig. 3.3: FCFS Scheduling

- To determine the number of head movements you would simply find the number of tracks it took to move from one request to the next.

  Total head Movement = (95-50)+(180-95)+(180-34)+(119-34)+(119-11)+
  (123-11)+(123-62)+(64-62)
  = 45+85+146+85+108+112+61+2
  = 644

- The disadvantage of this algorithm is noted by the oscillation from track 50 to track 180 and then back to track 11 to 123 then to 64. As we will soon see, this is the worst algorithm that one can use.

Advantages:

1. Every request gets a fair chance.
2. No indefinite postponement.

Disadvantages:

1. Algorithm does not try to optimize seek time.
2. It may not provide the best possible service.

3.2.2.2 SSTF

- The SSTF (Shortest Seek Time First) algorithm selects the request with the minimum seek time from the current head position.
- In other words, SSTF chooses the pending request closest to the current head position. SSTF scheduling is essentially a form of Shortest-Job-First (SJF) scheduling; and like SJF scheduling.
- So, the seek time of every request is calculated in advance in the queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first.
- For example, consider a disk queue with requests for I/O to block on cylinders 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the cylinder 50 and the tail cylinder being at 199.
- In this case the request is serviced according to the next shortest distance. Starting at 50, the next shortest distance would be 62 instead of 34 since it is only 12 tracks away from 62 and 16 tracks away from 34. The process would continue until all the processes are taken care of.
- For example, the next case would be to move from 62 to 64 instead of 34 since there are only 2 tracks between them and not 18 if it were to go the other way.

  Total Head Movement = (64 – 50) + (64 – 11) + (180 – 11) = 14 + 53 + 169 = 236 tracks.

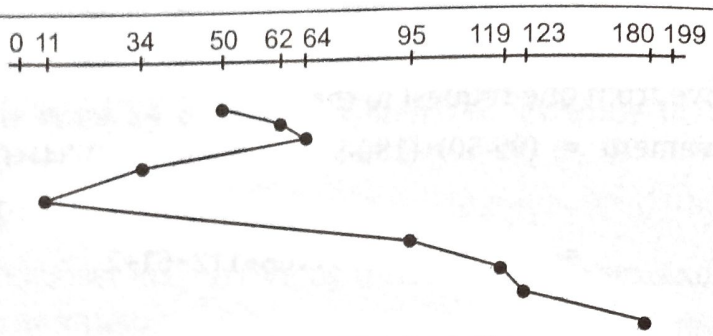

Fig. 3.4: SSTF Scheduling

Advantages:

1. It decreases average response Time.
2. It increases throughput.

Disadvantages:

1. Overhead to calculate seek time in advance.
2. It can cause starvation for a request if it has higher seek time as compared to incoming requests.
3. High variance of response time as SSTF favors only some requests.

3.2.2.3 SCAN

- In the SCAN algorithm, the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of the disk, it reverses its direction and again services the request arriving in its path.
- So, this algorithm works like an elevator and hence is also known as elevator algorithm. It moves the Read/Write (R/W) head back-and-forth to the innermost and outermost track.
- As it scans the tracks from end to end, it processes all the requests found in the direction it is headed.
- This will ensure that all track requests, whether in the outermost, middle or innermost location, will be traversed by the access arm thereby finding all the requests.
- For example, consider a disk queue with requests for I/O to block on cylinders 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the cylinder 50 and the tail cylinder being at 199. Assume direction towards 0.

  Total Head Movement = (50-0) + (180-0) = 50 + 180 = 230.

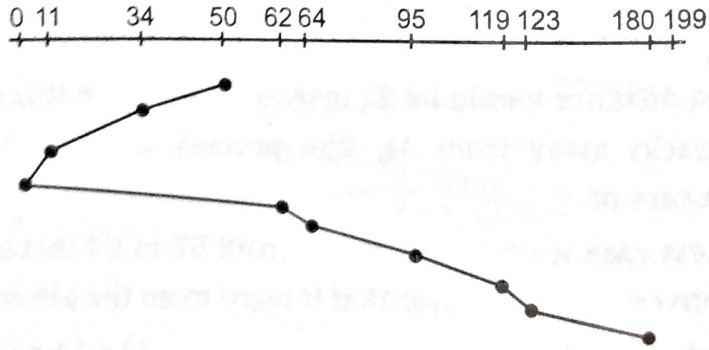

Fig. 3.5: SCAN Scheduling

Advantages:

1. It gives high throughput.
2. It provides low variance of response time.
3. It provides average response time.

Disadvantages:

1. Long waiting time for requests for locations just visited by disk arm.

C-SCAN:

- In the SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction.
- So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area. These situations are avoided in the C-SCAN algorithm.
- In C-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.
- It treats the cylinders as a circular list that wraps around from the last cylinder to the first one.
- For example, consider a disk queue with requests for I/O to block on cylinders 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the cylinder 50 and the tail cylinder being at 199. Assume direction towards 0.

Total Head Movement = (50 – 0) + (199 – 62) = 50 + 137 = 187.

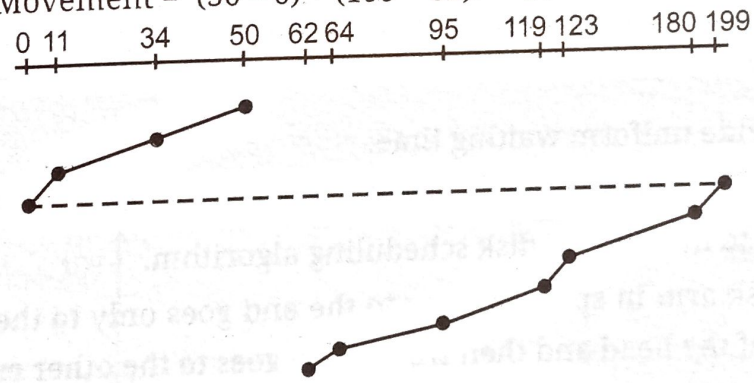

Fig. 3.6: C-SCAN Scheduling

Advantage: Provides more uniform wait time compared to SCAN.

3.2.2.4 LOOK

- It is similar to the SCAN disk scheduling algorithm except the difference is that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only.
- Thus, it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

- The disk arm starts at the first I/O request on the disk, and moves toward the last I/O request on the other end, servicing requests until it gets to the other extreme I/O request on the disk, where the head movement is reversed and servicing continues.
- It moves in both directions until both last I/O requests; more inclined to serve the middle cylinder requests.
- For example, consider a disk queue with requests for I/O to block on cylinders 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the cylinder 50 and the tail cylinder being at 199. Assume direction towards 0.

Total head Movement = (50 – 11) + (180 – 11)

= 39 + 169 = 208

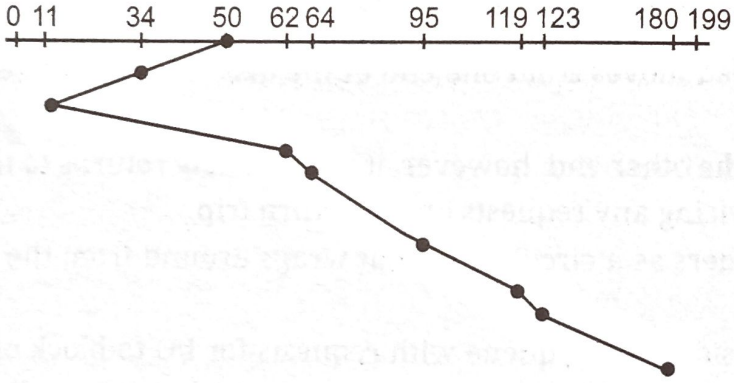

Fig. 3.7: LOOK Scheduling

Advantages:

1. Better than the SCAN algorithm, in terms of head movement.

Disadvantages:

1. It doesn't provide uniform waiting time.

C-LOOK:

- C-LOOK is similar to the C-SCAN disk scheduling algorithm.
- In C-LOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request.
- Thus, it also prevents the extra delay which occurred due to necessary traversal to the end of the disk.
- In general, Circular versions are fairer but pay with a larger total seek time. Scan versions have a larger total seek time than the corresponding Look versions.
- For example, consider a disk queue with requests for I/O to block on cylinders 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the cylinder 50 and the tail cylinder being at 199. Assume direction towards 0.

Total Head Movement = (50 – 11) + (180 – 62) = 39 + 118 = 157 tracks

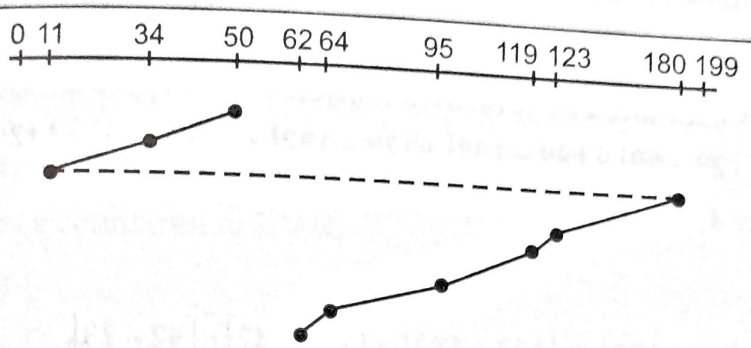

Fig. 3.8: C-Look Scheduling

- When selecting a Disk Scheduling algorithm, performance depends on the number and types of requests. SSTF is common and has a natural appeal. SCAN gives better performance than FCFS and SSTF.
- 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. Either SCAN or C-LOOK is a reasonable choice for the default algorithm.

Solved Problems

Problem 1: Consider Work Queue: 23, 89, 132, 42, 187 and schedule by algorithms:

1. FCFS (FIFO)
2. SSTF
3. SCAN
4. LOOK
5. C-SCAN
6. C-LOOK

There are 200 cylinders numbered from 0-199. The disk head starts at number 100.

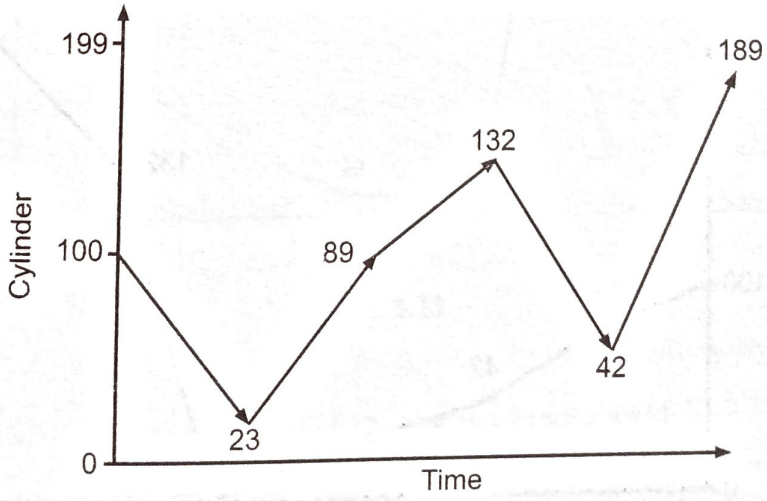

Fig. 3.9

1. FCFS (FIFO):

Total time is estimated by total arm motion

$|100 - 23| + |23 - 89| + |89 - 132| + |23 - 132| + |132 - 42| + |42 - 187|$

$= 77 + 66 + 43 + 109 + 90 + 145$

$= 530$

2. SSTF:

$|100 - 89| + |89 - 132| + |132 - 187| + |187 - 42| + |42 - 23|$

$= 11 + 43 + 55 + 145 + 19 = 273$

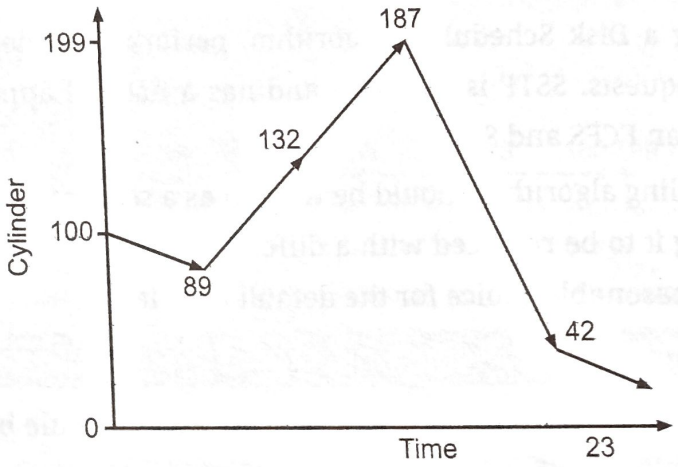

Fig. 3.10

3. SCAN:

Assume we are going inwards (i.e., towards 0).

$|100 - 89| + |89 - 42| + |42 - 23| + |23 - 0| + |0 - 132| + |132 - 187|$

$= 11 + 47 + 19 + 23 + 132 + 55$

$= 287$

Refer Fig. 3.11.

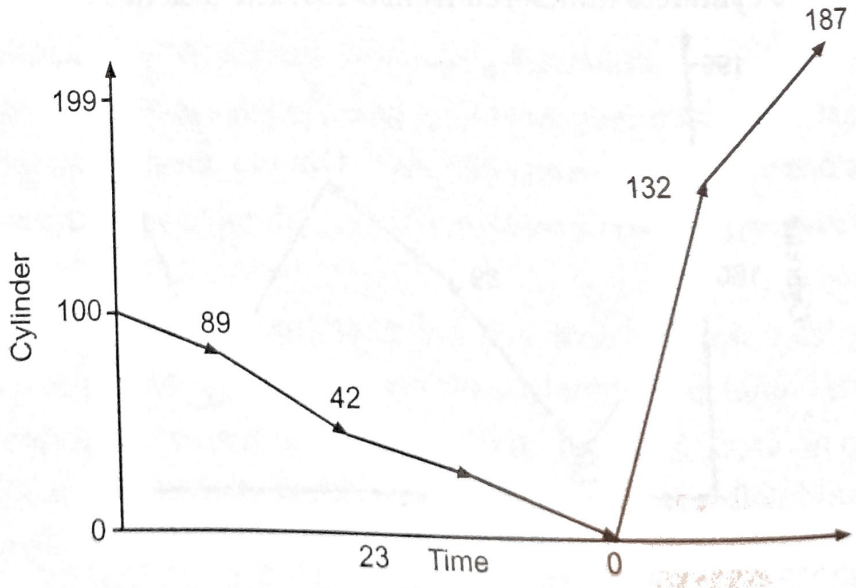

Fig. 3.11

4. **LOOK:**

$|100 - 89| + |89 - 42| + |42 - 23| + |23 - 132| + |132 - 187|$

$= 11 + 47 + 19 + 109 + 55 = 241$

Reduce variance compared to SCAN.

Refer Fig. 3.12.

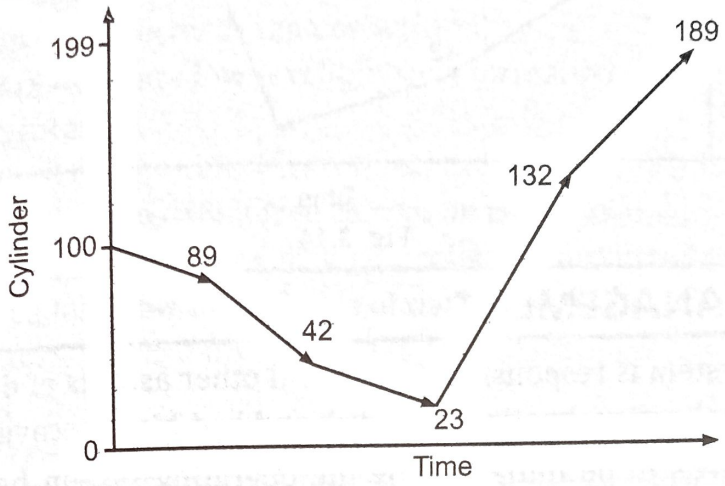

Fig. 3.12

5. **C-SCAN:**

$|100 - 89| + |89 - 42| + |42 - 23| + |23 - 0| + |0 - 199| + |199 - 187| + |187 - 132|$

$= 11 + 47 + 19 + 23 + 199 + 12 + 55 = 366$

Refer Fig. 3.13.

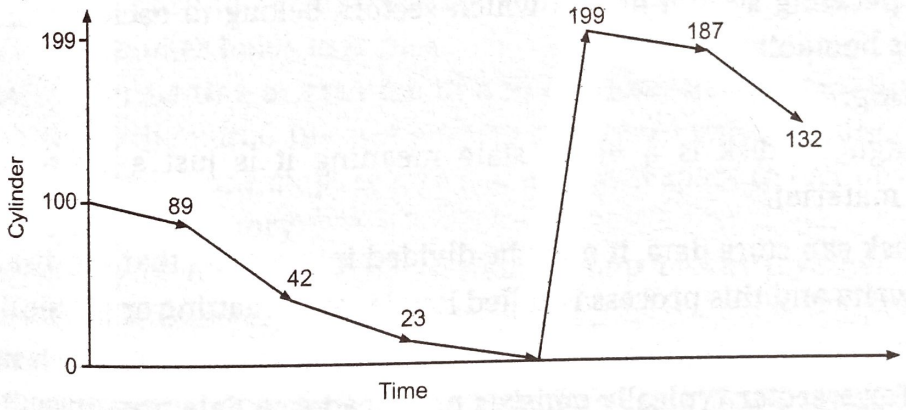

Fig. 3.13

6. **C-LOOK:**

$|100 - 89| + |89 - 42| + |42 - 23| + |23 - 187| + |187 - 132|$

$= 11 + 47 + 19 + 164 + 55 = 296$

Refer Fig. 3.14.

- If the stored and calculated numbers are different, this mismatch indicates that the data area of the sector has become corrupted and that the disk sector may be bad.
- The ECC contains enough information, if only a few bits of data have been corrupted, to enable the disk controller to identify which bits have changed and calculate what their correct values should be.
- ECC then reports recoverable soft errors. The disk controller automatically does the ECC processing whenever a sector is read or written.
- For a numbers of hard disks are low-level/physical formatted at the factory as a part of the manufacturing process.
- This formatting enables the manufacturer to test the disk and to initialize the mapping from logical block numbers to defect-free sectors on the disk.
- For number of hard disks, when the disk controller is instructed to low-level-format the disk, it can also be told how many bytes of data space to leave between the header and trailer of all sectors. It is usually possible to choose among a few sizes, such as 256, 512 and 1,024 bytes.
- Formatting a disk with a larger sector size means that fewer sectors can fit on each track; but it also means that fewer headers and trailers are written on each track and more space is available for user data. Some operating systems can handle only a sector size of 512 bytes.
- Before it can use a disk to hold files, the operating system still needs to record its own data structures on the disk. It does so in following two steps:

  Step 1: To partition the disk into one or more groups of cylinders. The operating system can treat each partition as though it were a separate disk. For example, one partition can hold a copy of the operating system's executable code, while another holds user files.

  Step 2: The logical formatting or creation of a file system. In this step, the operating system stores the initial file-system data structures onto the disk. These data structures may include maps of free and allocated space (a FAT or inodes) and an initial empty directory.
- To increase efficiency, a number of file systems group blocks together into larger chunks, frequently called clusters.
- Disk I/O is done via blocks, but file system I/O is done via clusters; effectively assuring that I/O has more sequential-access and fewer random-access characteristics.
- Some operating systems give special programs the ability to use a disk partition as a large sequential array of logical blocks, without any file-system data structures. This array is sometimes called the raw disk, and I/O to this array is termed raw I/O.
- For example, some database systems prefer raw I/O because it enables them to control the exact disk location where each database record is stored.
- Raw I/O bypasses all the file-system services, such as the buffer cache, file locking, pre-fetching, space allocation, file names, and directories.

- We can make number of applications more efficient by allowing them to implemen their own special-purpose storage services on a raw partition, but most application perform better when they use the regular file-system services.

Boot Block:

- For a computer to start running - for example, when it is powered up - it must have a initial program to run known as bootstrap program.
- The bootstrap program initializes all aspects of the system, from CPU registers t device controllers and the contents of main memory and then starts the operatin system.
- To do its job, the bootstrap program finds the operating-system kernel on disk, load that kernel into memory, and jumps to an initial address to begin the operating-syste execution.
- For a number of computers, the bootstrap is stored in ROM (Read Only Memory). This location is convenient, because the ROM needs no initialization and is at a fixed location that the processor can start executing when powered up or reset.
- And since ROM is read only, it cannot be infected by a computer virus. The problem is that changing this bootstrap code requires changing the ROM hardware chips.
- For this reason, number of systems stores 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 system disk or boot disk.
- The code in the boot ROM instructs the disk controller to read the boot blocks into memory (at this point no device drivers are loaded) and then starts executing that code.
- The full bootstrap program is more sophisticated than the bootstrap loader in the boot ROM; it is able to load the entire operating system from a non-fixed location on disk and to start the operating system running. Even so, the full bootstrap code may be small.
- Fig. 3.15 shows the boot process in the Windows 2000 system with its boot code in the first sector on the hard disk (which it terms the MBR (Master Boot Record).
- Windows 2000 system allows a hard disk to be divided into one or more partitions; one partition, identified as containing the operating system and device drivers.
- Booting begins in Windows 2000 by running code that is resident in the system's ROM memory and this code directs the system to read the boot code from the MBR.
- In addition to containing boot code, the MBR contains a table listing the partitions for the hard disk and a flag indicating which partition the system is to be booted from, as shown in Fig. 3.15.

- Once the system identifies the boot partition, it reads the first sector from that partition (which is called the boot sector) and continues with the remainder of the boot process, which includes loading the various subsystems and system services.

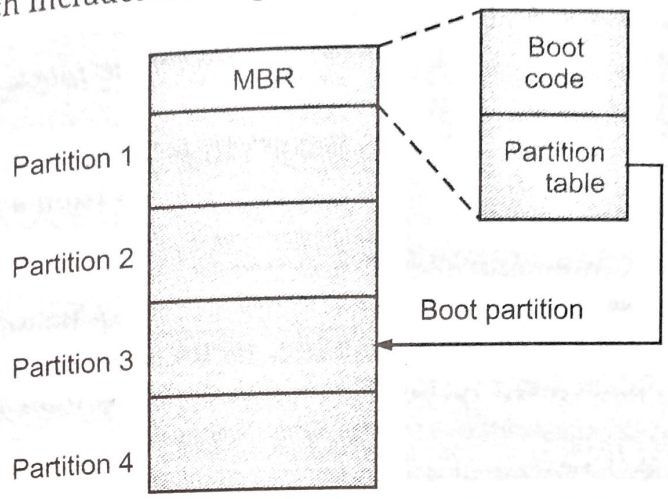

Fig. 3.15: Booting from Disk in Windows 2000 System

Bad Blocks:

- Because disks have moving parts and small tolerances, they are prone to failure. Sometimes the failure is complete; in this case, 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.
- On simple disks, like some disks with IDE (Integrated Development Environment) controllers, bad blocks are handled manually.
- For example, 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 (File Allocation Table) entry to tell the allocation routines not to use that block.
- If blocks go bad during normal operation, a special program like 'chkdsk' must be run manually to search for the bad blocks and to lock them away. Data that resided on the bad blocks usually are lost.
- The SCSI (Small Computer System Interface) disks used in high-end PCs and most workstations and servers, are smarter about bad-block recovery.
- The disk controller maintains a list of bad blocks on the disk. The list is initialized during the low-level/physical formatting at the factory and is updated over the life of the disk.
- Physical formatting also sets aside spare sectors not visible to the operating system. The disk controller can be told to replace each bad sector logically with one of the spare sectors known as forwarding or sector sparing.

- An alternative to sector some controllers can be instructed to replace a bad block by sector slipping.
- The replacement of a bad block generally is not totally automatic because the data in the bad block are usually lost.
- Soft errors may trigger a process in which a copy of the block data is made and the block is spared or slipped.
- An unrecoverable hard error, however, results in lost data. Whatever file was using that block must be repaired (for example, by restoration from a backup tape) and that requires manual intervention.
- Depending on the disk and controller in use, these blocks are handled in a variety of ways such as sector sparing or forwarding (controller can be told to replace each bad sector logically with one of the spare sectors), sector slipping (all sectors between the bad sector and the replacement sector are moved down by one) and so on.