Disk Performance Optimization

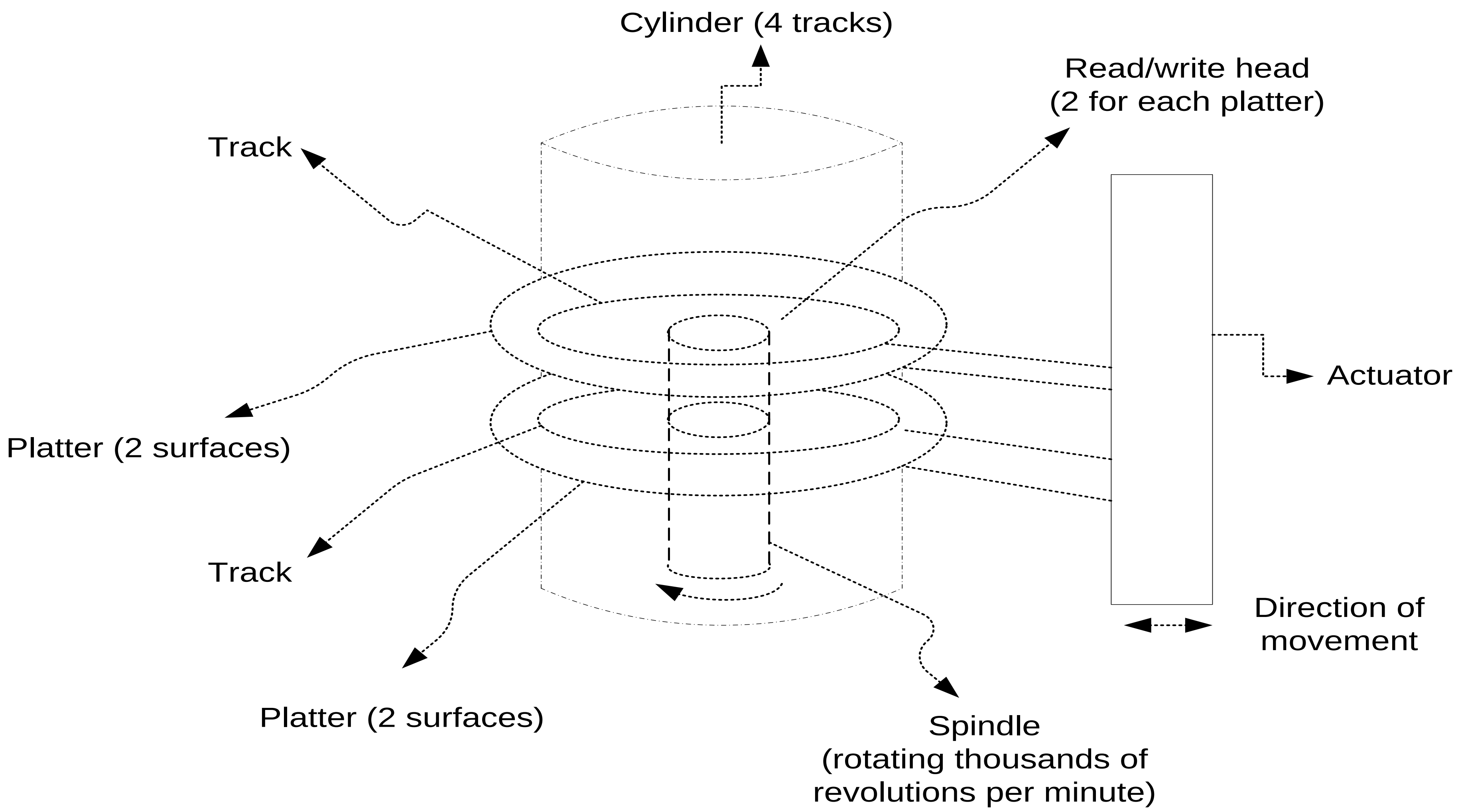
By

Lecturer: Ameen A.Noor

Introduction

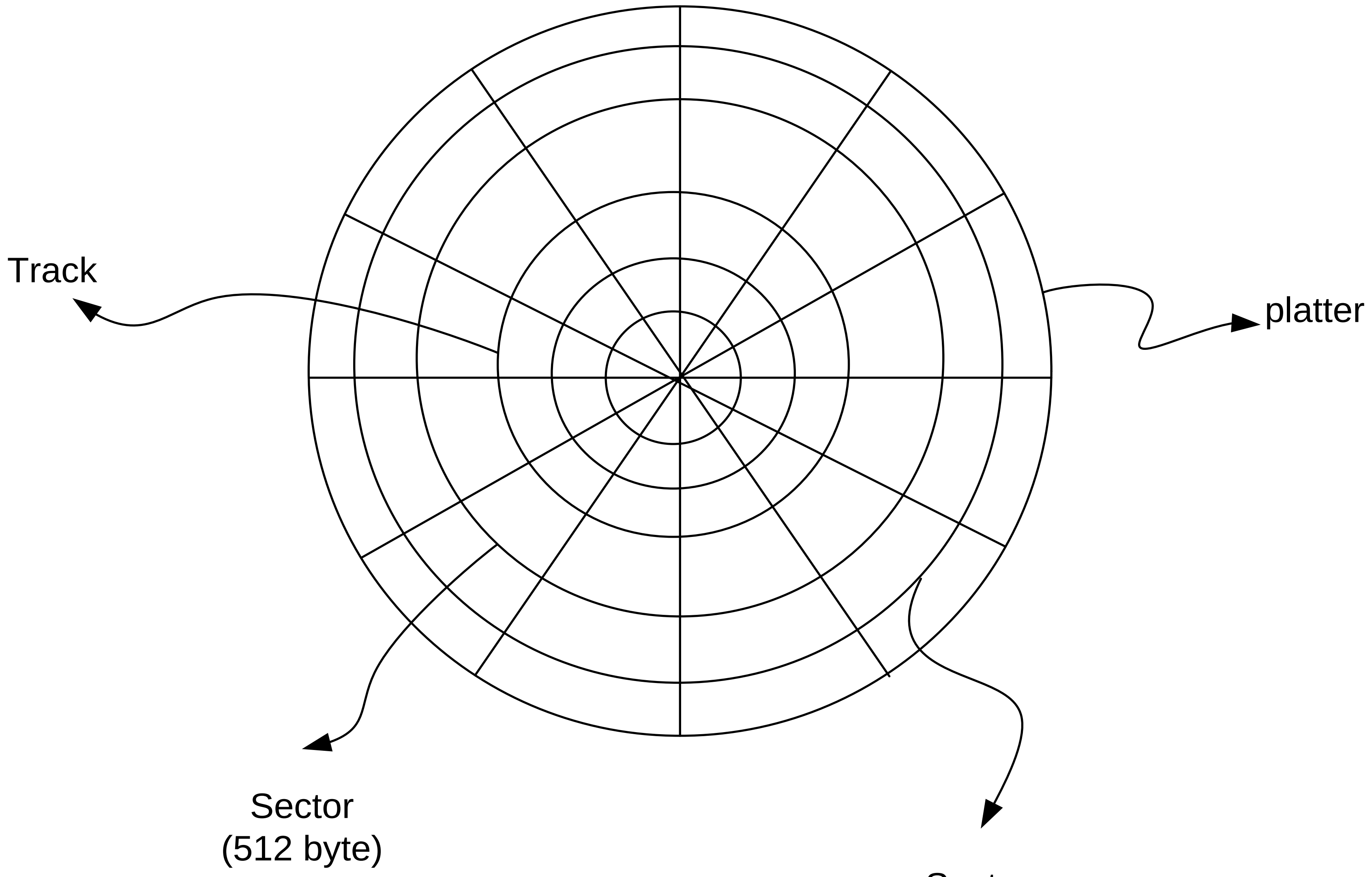
- processors & memory speeds have increased more rapidly than those of hard disk.
- processes requesting data from disk tend to experience long service delay.
- we discuss how to optimize disk performance by recording disk requests to increase throughput, decrease response time & reduce the variance of response times. We also discuss how OSs reorganize data on disk & exploit buffers & caches to boost performance.
- Finally, we discuss Redundant Arrays of Independent Disks (RAIDs), which improve disk access times & fault tolerance by servicing requests using multiple disks at once.

general structure of hard disk



The disk storage may consist of several platters & each has a separate read/write moving-head. All heads are fixed to the same actuator & hence move together to select certain cylinder. The cylinder is a set of tracks on all surfaces. Usually, at one time, only one head is active & deals with one track of the whole cylinder. This means that OS has to select the proper head to read/write (r/w) data.

Each track is divided to several sectors as shown in fig 10.2 each sector is of 512 byte size.



Tracks & Sectors of Disk

it is clear that for the OS to R/W data from disk, it needs to:

1- Specify the proper surface containing the data & hence the proper moving head.

2- Specify the track & sectors containing the data on that surface.

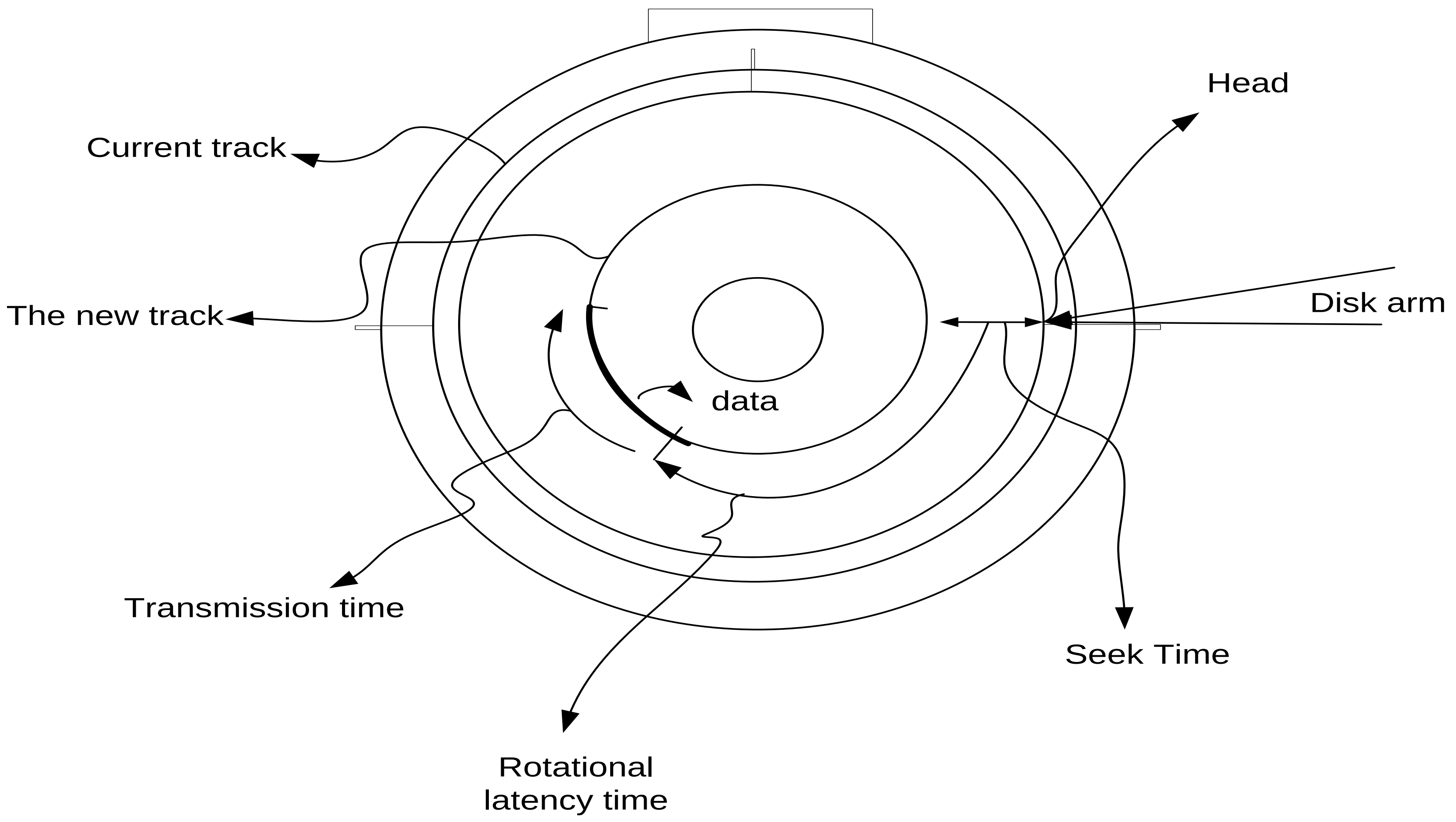
3- Instruct actuator to move head to the proper track. This movement takes time which is called "Seek Time" & its average value is in the range of few milliseconds (e.g. 7 msec).

4-The platter has to be rotating & the head should wait for the proper sector in the track to get the data.

This time depends on revolution speed & its average value is half of one revolution period & is usually of few milliseconds value (e.g. 4 msec). This time is called "Latency Time".

5- When the head is on the proper sector, it starts reading/writing data & also this process takes time depending on number of sectors to be read.

It is clear that a few milliseconds are necessary to R/W data from the disk while the CPU can execute millions of instructions in thatv time.



components of disk access

Disk Capacity

Capacity: maximum number of bits that can be stored

- Vendors express capacity in units of gigabytes (GB), where 1 GB = 10^9 Bytes (Lawsuit pending! Claims deceptive advertising)

Capacity is determined by these technology factors:

- Recording density (bits/in): number of bits that can be squeezed into a 1 inch linear segment of a track
- Track density (tracks/in): number of tracks that can be squeezed into a 1 inch radial segment
- Areal density (bits/in^2): product of recording and track density

Computing Disk Capacity

Capacity = (# bytes/sector) x (avg. # sectors/track) x

(# tracks/surface) x (# surfaces/platter) x

(# platters/disk)

Example:

- 512 bytes/sector
- 1000 sectors/track (on average)
- 20,000 tracks/surface
- 2 surfaces/platter
- 5 platters/disk

Capacity = 512 x 1000 x 80000 x 2 x 5

= 409,600,000,000

= 409.6 GB

Why Disk Scheduling is Necessary

Many processes can generate requests for reading & writing data on a disk simultaneously. Because these processes sometimes makes requests faster than they can be serviced by the disk, waiting lines or queues build up to hold disk requests. Some early computing systems simply serviced these request on a "First Come First Served FCFS" basis, in which the earliest arriving request is serviced first. FCFS exhibits a random seek pattern in which successive requests can cause time consuming seeks from the innermost to the outermost cylinders (tracks). To reduce the time spent seeking records, it seems reasonable to reorder the request queue in some manners other than FCFS. This process, called disk scheduling, can significantly improve throughput

The two most common types of scheduling are "Seek optimizing" & "Rotational Optimizing". Because seek times are usually greater than latency times, most scheduling algorithms concentrate on minimizing total seek time for a set of requests.

Disk Scheduling Strategies

- The strategies are evaluated by the following criteria:
- - Throughput: The number of requests serviced per unit time. The maximum number is the better.
- - Mean response time: The average time spent waiting for a request to be serviced. The minimum time is the better.
- - Variance of response time: The difference between the request waiting time andthe mean response time. The minimum variance is the better.

1 First Come First Served (FCFS) Disk Scheduling

This has been already discussed & it suffers from long seek time & hence low throughput especially under heavy loads.

2 Shortest Seek Time First (SSTF)

Ithe next request to be serviced is the one that is closest to the R/W head & thus incurs the shortest seek time.

The main problem is the possibility of indefinite postponement for the innermost & outermost tracks especially under heavy loads i.e. many requests are coming all the time.

3 Scan Disk Scheduling

Here, the disk head moves from the outer track to the inner & then in the opposite direction. The request to be serviced is the one that its track is ahead of the head in the motion direction.

This means that the requests coming in front of the head in the motion direction are serviced first.

The scheduling may suffer indefinite postponement or long waits for requests of innermost and outermost tracks under heavy load.

4 C-Scan Disk Scheduling

C-Scan mean circular scan & it is similar to SCAN but the head doesn't service requests when moving in the opposite direction i.e. it service requests in only one direction & hence decrease the possibility of indefinite postponement of outerside tracks.

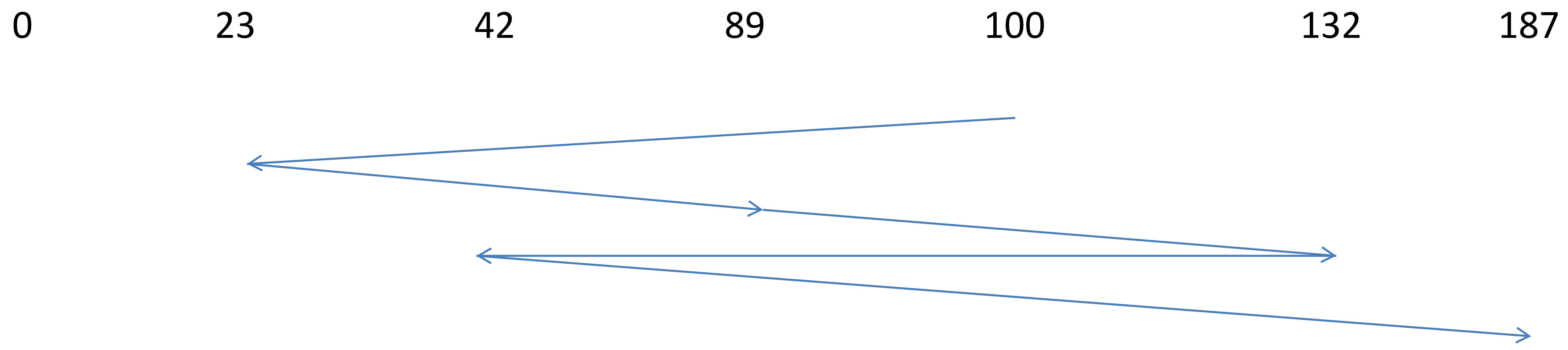
5 Other scheduling strategies

There are also other strategies such as:

- ❑ Fscan
- ❑ N-Step Scan
- ❑ Look Scan
- ❑ C-Look Scan
- ❑ Shortest Latency Time First (SLTF)
- ❑ Shortest Positioning Time First (SPTF)
- ❑ Shortest Access Time First (SATF)

Ex: a disk queue with requests for I/O to blocks on cylinders
23, 89, 132, 42, 187 With disk head initially at 100

1- FCFS 23, 89, 132, 42, 187

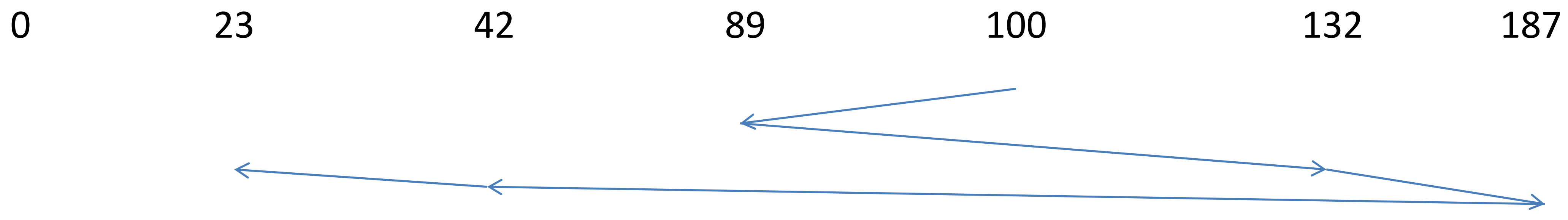


77+66+43+90+145=421

If the requests for cylinders 23 and 42 could be serviced together, total head movement could be decreased substantially.

2- SSTF

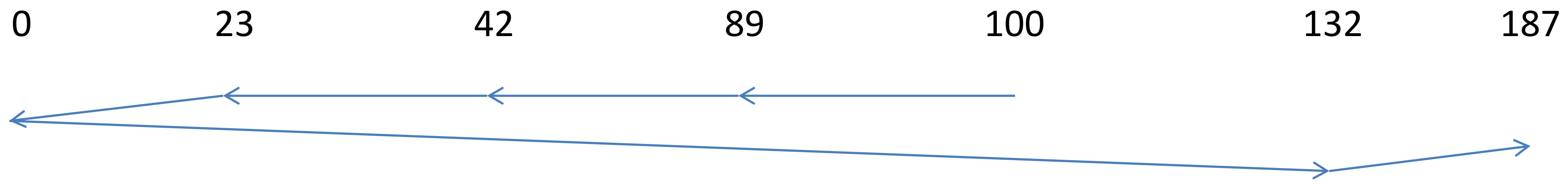
23, 89, 132, 42, 187



11+43+55+145+19=273

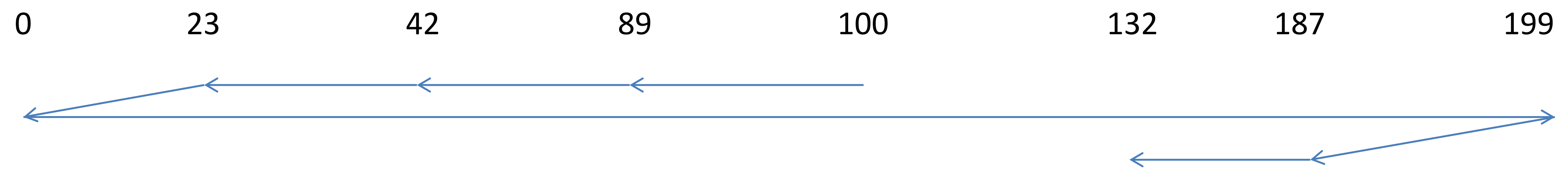
3- SCAN

23, 89, 132, 42, 187



11+47+19+23+132+55=287

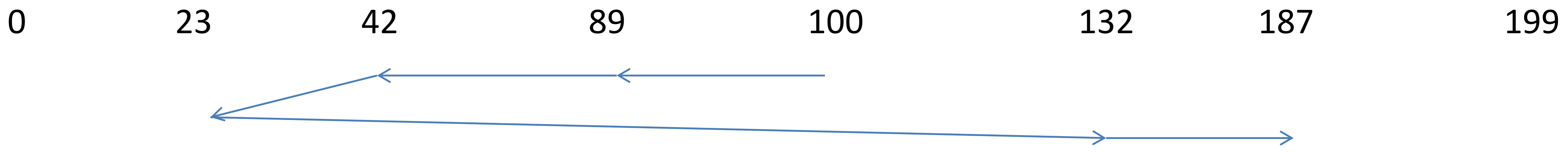
4- C-SCAN 23, 89, 132, 42, 187



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

Head movement can be reduced if the request for cylinder 187 is serviced directly after request at 23 without going to the disk 0

5- LOOK 23, 89, 132, 42, 187



11+47+19+109+55=3241

Compared to SCAN, LOOK saves going from 23 to 0 and then back. Most efficient for this sequence of requests

Caching & Buffering

- Many systems maintain a "disk cache buffer", which is a region of main memory that the OS reserves for disk data. In one context, the reserved memory acts as cache, allowing processes quick access to data that would otherwise need to be fetched from disk. The reserved memory also acts as a buffer, allowing the OS to delay writing modified data until the disk experiences a light load or until the disk head is in a favorable position to improve I/O performance.

- The disk cache buffer presents several challenges to OS designers such as:
  - Size of cache buffer
  - Replacement strategy
  - Inconsistency of data when power or system fail.

Many of today's hard disk drives maintain an independent high-speed buffer cache (on board cache) of several megabytes it's not related to main memory i.e. not part of it (i.e. can't be addressed by CPU directly).
Also, some hard disk controllers (e.g. SCSI, RAID) maintain their own buffer cache (normal RAM) separate from main memory.
ALL buffers are used to enhance the disk performance i.e. increase the speed of data retrieval.

Redundant Arrays of Independent Disks (RAID)

- The disk includes several platters. Each platter has two R/W heads. ALL the heads are mounted on one actuator & hence move together.
- the OS determines the location of data on which surface of which platter & instructs the proper head to R/W.
- At any one time, only one head is used for reading or writing i.e. it is not possible to make multiple accesses with several heads.

In other words, the disk has multiple heads but only one of them is used at any one time.

- The file is usually stored on one surface of one platter only unless it is very large.
- The only objective of this disk structure is to get large storage volume.

In the RAID structure, the philosophy is completely different from the nonRAID as it has the following features:

- The disk includes several platters & heads as before but each head here has its own actuator and hence can move independently of the other heads. This will enable multiple reads & writes to be carried out at the same time & hence faster disk response.

- The file may be stored on one platter or on several ones & hence it is possible to read/write several parts of the same file at the same time & this means fast R/W.

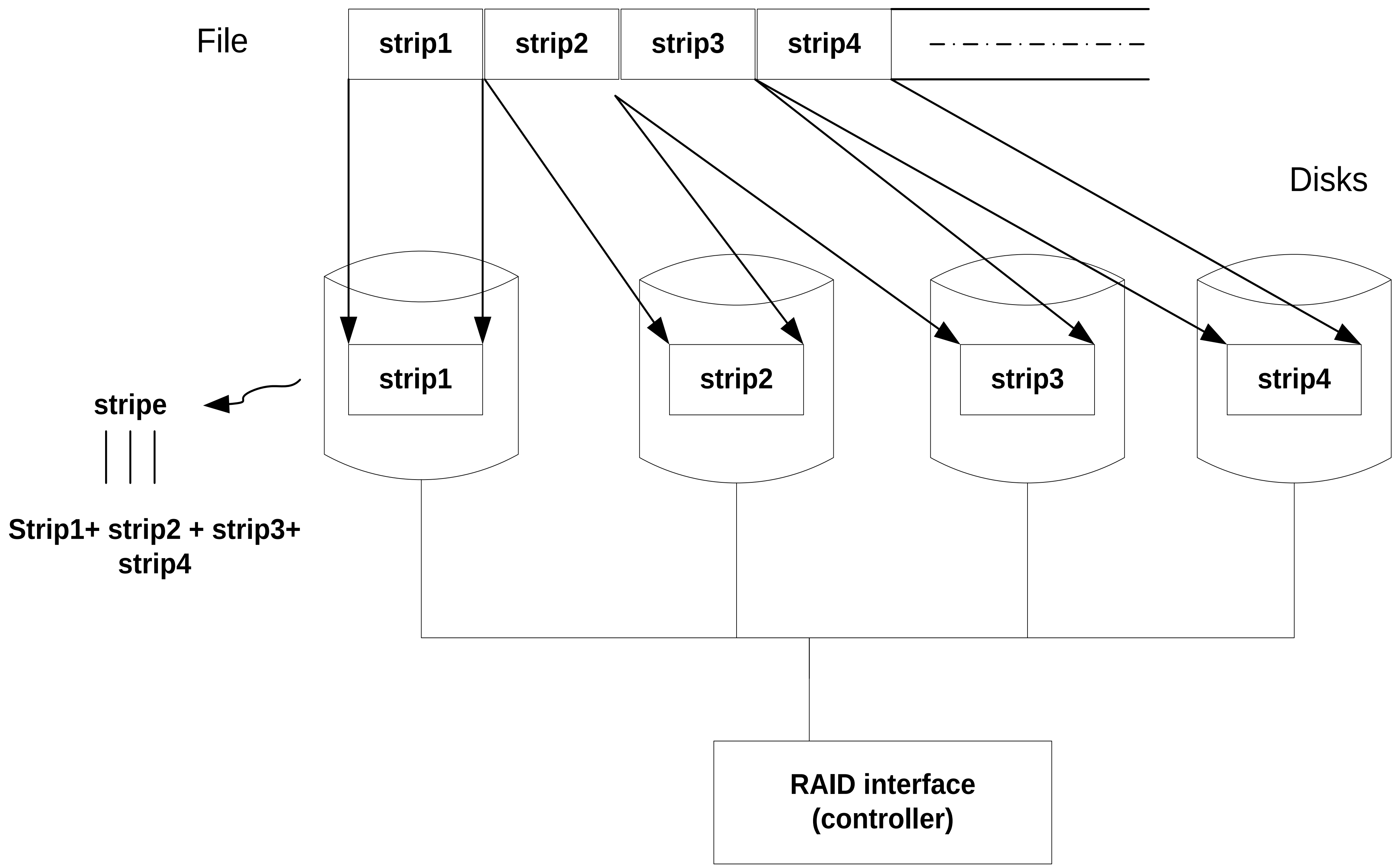
- Reliability issue is handled here and hence we find out that an error correcting code (ECC) is being used & hence more storage is needed and this is reason for the term "Redundancy". The redundancy will help in correcting errors & hence the RAID system will be "fault tolerant" in this case.

In RAID systems we use the following terms:
Data Striping: entail dividing data into fixed size blocks called "strips". Contiguous strips of a file are typically placed on separate disks so that request for file data can be serviced using multiple disks at once, which improves access times.

Stripe: consists of the set of strips at the same location on each disk of the array.

Fine grained strips: small size strips & this tend to spread file data across several disks & hence reduce access time.

Coarse grained strips: large size strips & this enable some files, to fit entirely on one strip & hence the access time is as in the Non-RAID system, for that file, however, several requests for several files can be serviced together (Simultaneously).



Strips & stripe in RAID systems

1-RAID Level :uses striped disk array with no fault tolerance and hence has no redundancy. The disk contains data only & there is no ECC data.
In level multiple reads &writes are possible.
The striping level (size) is a block.
Level is sometimes not considered as RAID as it has no redundancy

2 -RAID Level1 (Mirroring)

This level employs disk mirroring (shadowing) to provide redundancy, so each disk in the array is duplicated. Stripes are not implemented in level1 & hence multiple access for the same file is only possible on reading but not on writing. On writing, the same data has to be written on both disks (the original & the mirror) but on reading, 2 different parts of the same file can be read at the same time from the original & mirror disks. The mirror technology enhances reliability & restricted multiple access but doubling the cost.

3- RAID Level2

RAID level2 arrays are striped at the bit level, so each strip stores one bit. This means that adjacent bits of file are stored on different disks. Level2 arrays are not mirrored, which reduces the storage overhead incurred by Level1.

The fault tolerance is achieved here by using hamming error correcting codes (hamming ECCs). The error code bits are stored on separate disks (parity disks).

Of course, each stripe of one bit size on data disks has a stripe of one bit size on parity disks. This means that each group of data bits has a corresponding group of ECC bits.

The clear problem here is that if the OS wants to write few bits of the group (stripe), it has to read all data stripes first & then modify the necessary data bits & then calculate the ECC bits & at last store the new data & ECC bits. This is called "read-modify-write" cycle

4 - RAID Level3

RAID level3 stripes data at the bit or byte level but use parity checks for fault tolerance instead of Hamming. In parity check, we use only one bit (even or odd parity) & this bit does not locate the place of error (as incase of Hamming) but indicates only its existence. When the error occurs, the OS will inform the user immediately who has to find out the erroneous disk and replace it. The data on the faulty disk can be regenerated automatically with the help of other data disks & the parity disk. The advantages of such system:

1- Large storage.
2- Fault tolerance with one extra disk (one parity disk).
3- Multiple access for one file is possible and hence fast access time.
Of course, multiple access for several files is not possible as any file request will occupy all of the disks.

5 - RAID Level4

RAID Level4 systems are striped using fixed size blocks (typically much larger than a byte) & use one disk for parity (even or odd parity). The difference with level3 is that the file may occupy fraction of disks & not all of them (remember the coarse grained stripes) & hence multiple requests for multiple files may be possible at the same time. Here, we should remember that when reading data from disks, it is not always necessary to read parity bits as these bits are stored not for error detection but mainly for error correction (of one bit – usually one disk may be faulty-).

6 - RAID Level5

RAID Level5 arrays are striped at the block level & a parity check (even or odd) is used like in level4. The difference with level4 is that the parity bits are not located on one disk but distributed throughout the arrays of disks. This means that disk1 carries parity bit1, disk2 carries parity bit2, & so on. The advantage of this level is that multiple writes (writes for several files) are possible because the parity bits are not stored on one disk as the case of level4

7 - Other RAID Levels

There are other levels such as:

RAID Level6

RAID Level10+1

RAID Level10

RAID Level10+3, 0+5, 50, 1+5, etc.