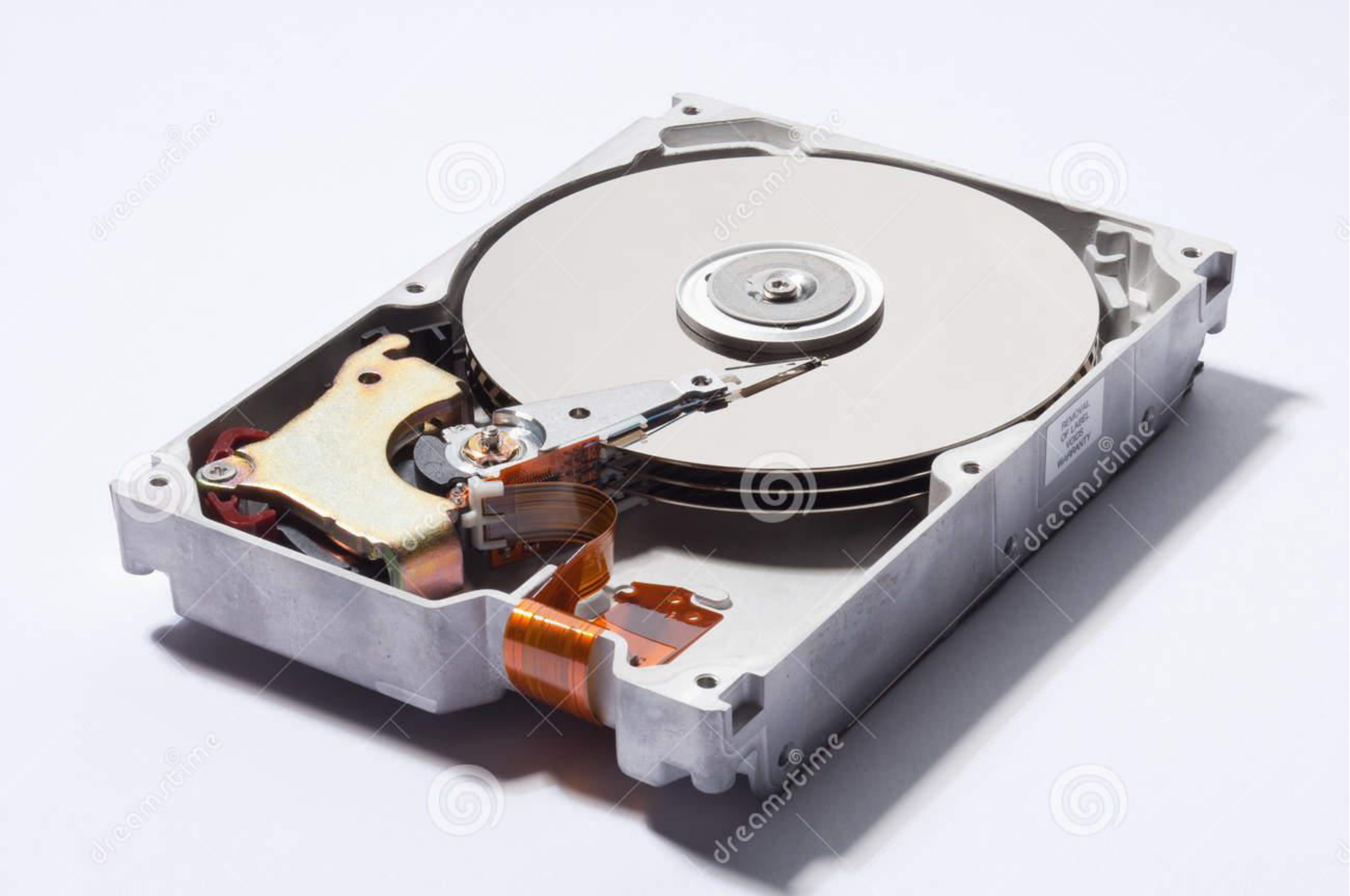
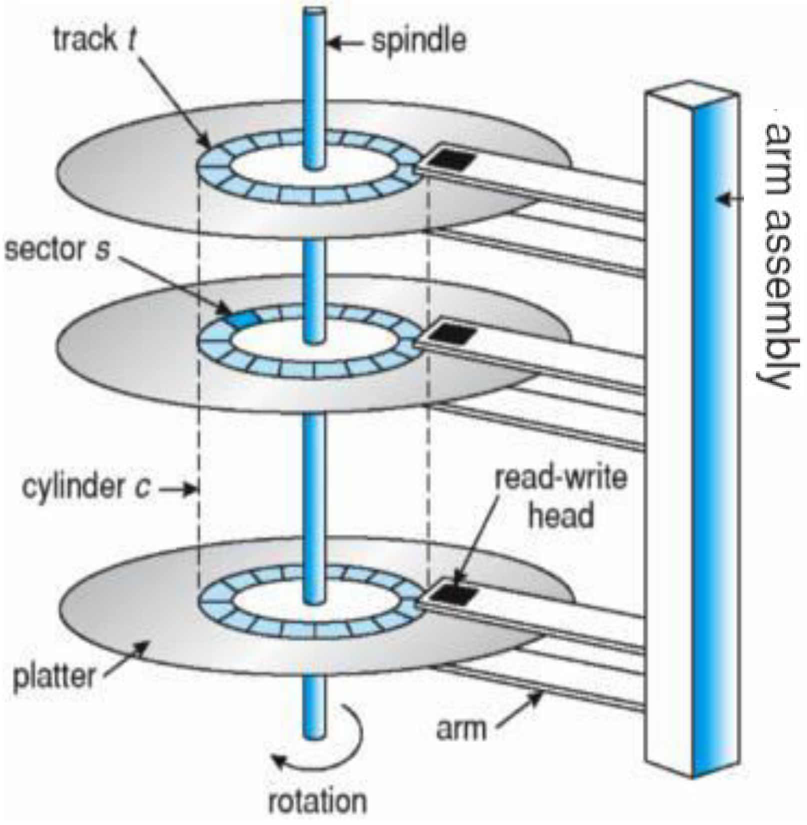
Mass Storage Structure

Magnetic disks



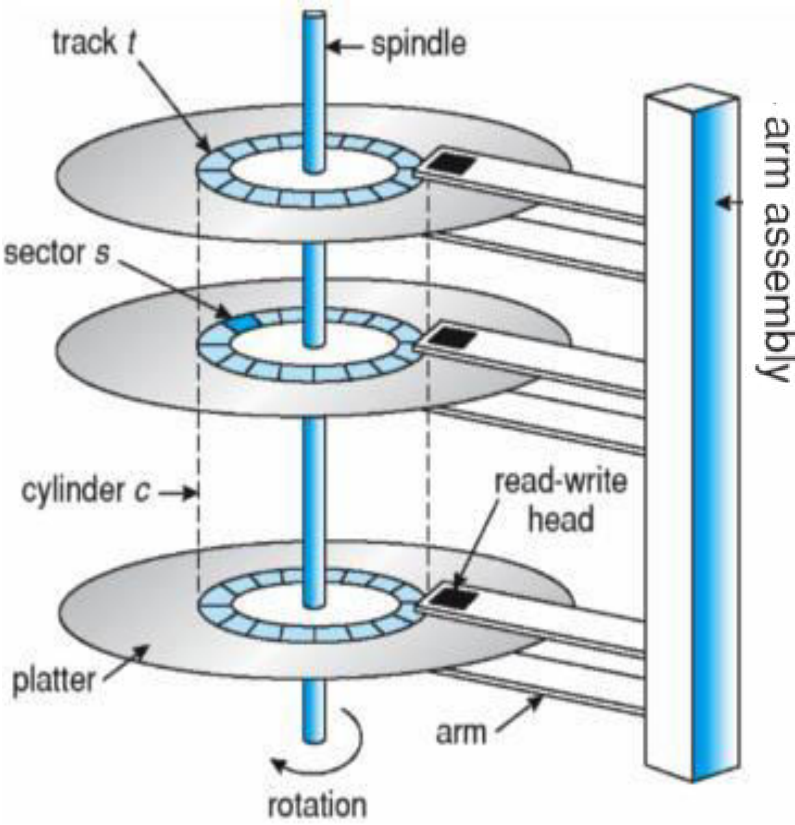
Magnetic disks

- *Magnetic disks provide the bulk of* ***secondary storage*** *for modern computer systems.*
- *Each disk platter has a* ***flat circular shape like CD****. Common platter diameter range from 1.8 to 3.5 inches.*
- *Both the surface of disk are covered with a* ***magnetic material****. Information is recorded magnetically on the platters.*
- *A* ***read-write head*** *flies just above each surface of every platter.*
- *The heads are attached to the disk arm.*

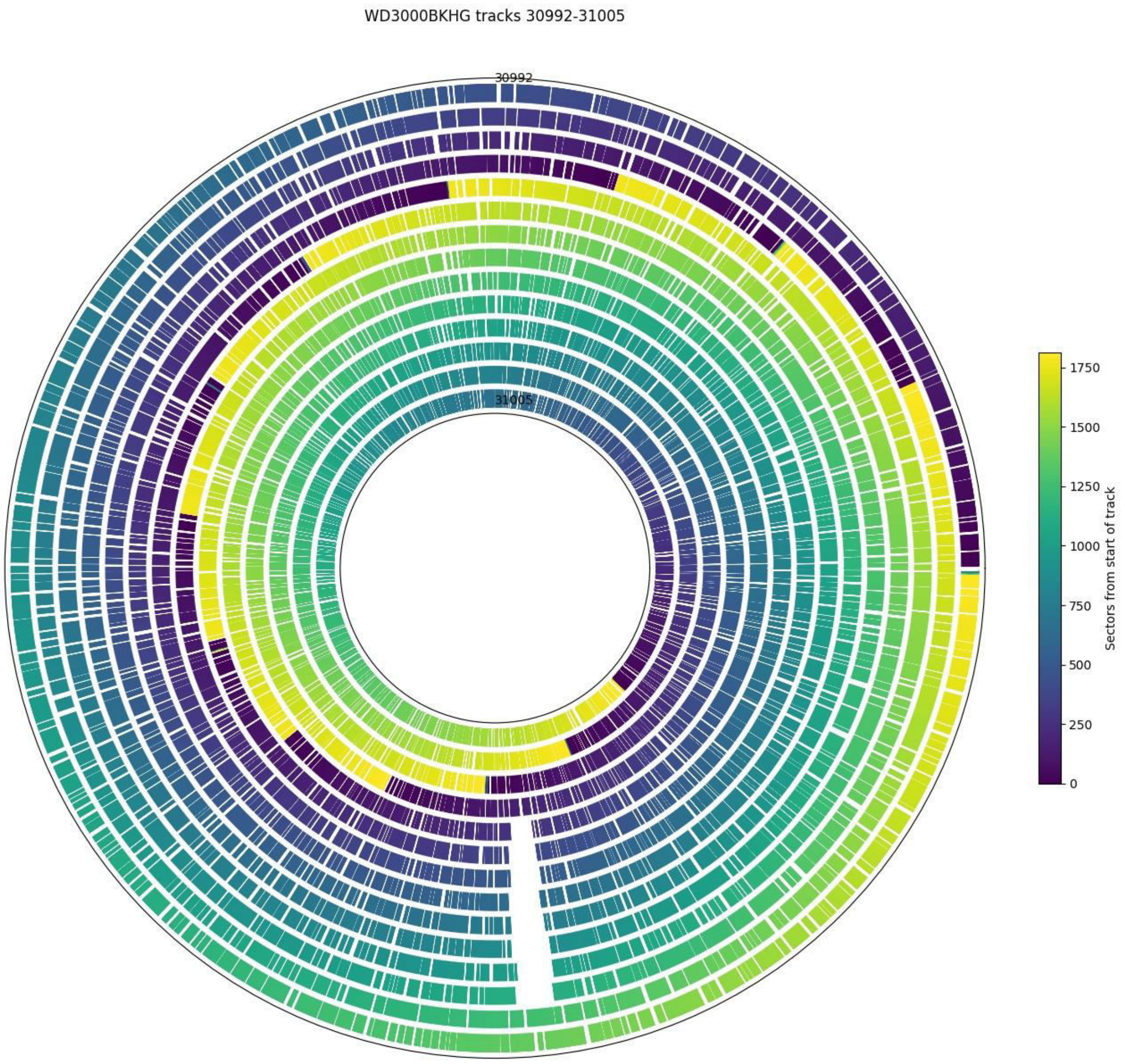


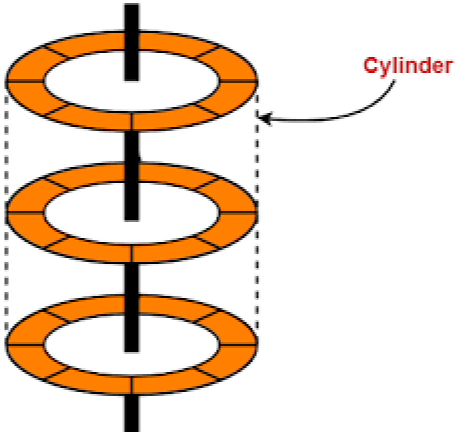
Magnetic disks

- *Each surface of platter is logically divided into circular **tracks**, which are subdivided into **sectors**.*
- *The collection of all tracks that are the same distance from the edge of the platter, (at one arm position i.e. all tracks **immediately above one another**) is called a **cylinder**.*



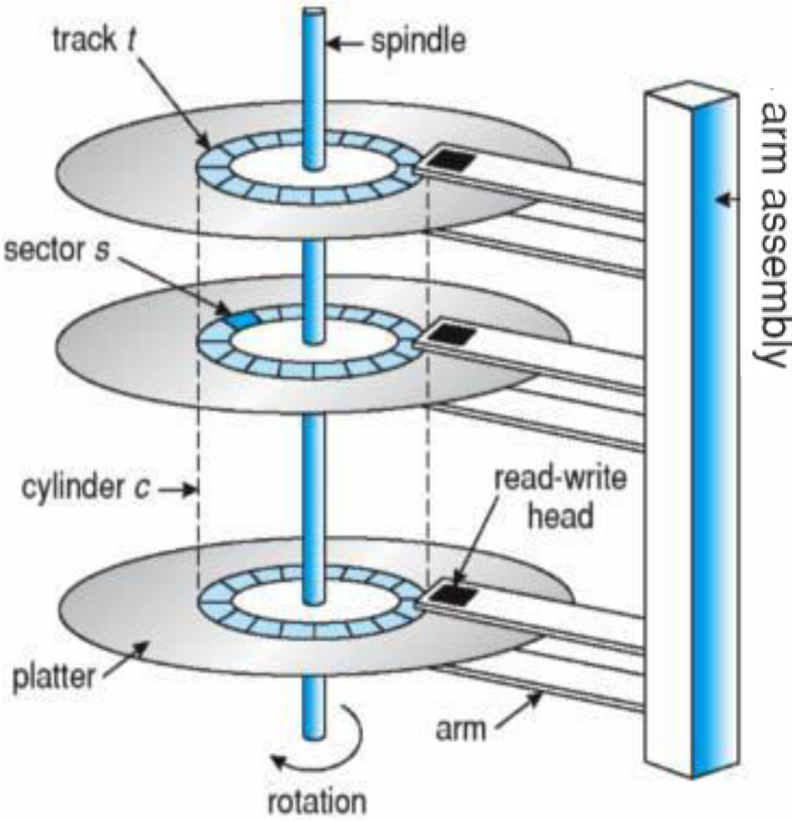
Magnetic disks





Magnetic disks

- *The data on a hard drive is read by read-write **heads**. The standard configuration uses one head per surface, each on a separate **arm**, and controlled by a common **arm assembly** which moves all heads simultaneously from one cylinder to another.*
- *There may be thousands of cylinders in a disk drive and each track may have hundred of sectors.*



Magnetic disks

- *Most drives rotates* ***60 to 250*** *times per minute specified in terms of rotations per minute* ***(RPM).***
- *The storage capacity of a traditional disk drive is equal to the number of* ***heads*** *(i.e. the number of working surfaces), times the number of* ***tracks*** *per surface, times the number of* ***sectors*** *per track, times the number of* ***bytes*** *per sector.*

Heads*tracks*sectors*byte in sector

Magnetic disks

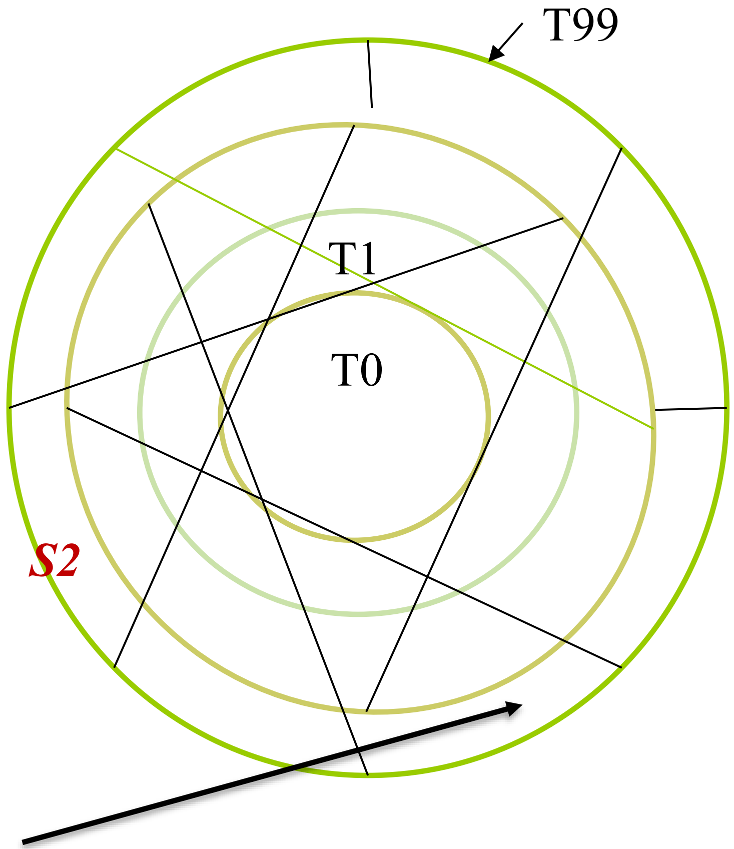
- *Disk heads "fly" over the surface, If they should **accidentally contact** the disk, then a **head crash** occurs, which may or may not permanently damage the disk or even destroy it completely.*
- *Disk drives are connected to the computer via a cable known as the **I/O Bus**. Several kinds of buses are available Advanced Technology Attachment (ATA), Serial ATA (SATA), Universal Serial Bus (USB); Fiber Channel (FC), and Small Computer Systems Interface (SCSI).*

Magnetic disks

- ***Seek time:*** *is the time taken to locate the disk arm to a specified track where the data is to be read or write.*
- ***Rotational Latency:*** *Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads.*
- ***Transfer rate:*** *which is the time required to move the data electronically from the disk to the computer.*

Disk Access Time = *Seek Time + Rotational Latency +Transfer Time*

Magnetic disks



Magnetic disks

Consider a disk pack with 16 surfaces, 128 tracks per surface and 256 sectors per track. 512 bytes of data are stored in a bit serial manner in a sector. The capacity of the disk pack and the number of bits required to specify a particular sector in the disk are respectively:

(A) 256 Mbyte, 19 bits

(B) 256 Mbyte, 28 bits

(C) 512 Mbyte, 20 bits

(D) 64 Gbyte, 28 bit

Magnetic disks

- $16 surface \times \frac{128 tracks}{surface} \times \frac{256 sectors}{track} \times \frac{512B}{sector} = 2^{28}B = 256MB$

- $16 surface \times \frac{128 tracks}{surface} \times \frac{256 sectors}{track} = 2^{19} sectors$

  $\therefore 19\ bits$

Solid-State Disks

- ***Solid state disks, or SSDs*** *used like HDD. More reliable than HDDs.*
- *SSDs is* ***nonvolatile memory*** *that is used as* ***a small fast hard drive****. Specific implementations may use either DRAM chips protected by a battery to sustain the information in power failure or the flash memory technologies like single level cell (SLC) and multilevel cell (MLC).*
- *Because SSDs have* ***no moving parts*** *they are* ***much faster than traditional hard drives****, and certain problems such as the scheduling of disk accesses simply do not apply (no seek time, no rotational latency).*
- *However SSDs also have their weaknesses: They are* ***more expensive*** *than* ***hard drives****, generally* ***not as large****, and may* ***have shorter life spans****.*

Solid-State Disks

- *SSDs are especially useful as* ***a high-speed cache*** *of hard-disk information that must be accessed quickly. One example is to store* ***filesystem meta-data****, e.g. directory and* ***inode*** *information, that must be accessed quickly and often. Another variation is a boot disk containing the OS and some application executables, but no vital user data. SSDs are also used in laptops to make them smaller, faster, and lighter.*
- ***Buses can be too slow:*** *connected directly to the system PCI (Peripheral Component Interconnect) for example.*

Magnetic Tapes

- ***Magnetic tapes*** *were once used for common secondary storage before the days of hard disk drives, but today are used primarily for backups.*
- *Accessing a particular spot on a magnetic tape can be slow, but once reading or writing commences, access speeds are comparable to disk drives.*
- *Capacities of tape drives can range from 20 to 200 GB, and compression can double that capacity.*

Magnetic Tape

- *Was early secondary-storage medium*
- *Relatively permanent and holds large quantities of data*
- *Access time slow- Random access ~1000 times slower than disk*
- *Mainly used for backup, storage of infrequently-used data, transfer medium between systems*
- *Once data under head, transfer rates comparable to disk*
  - *140MB/sec and greater*
- *200GB to 1.5TB typical storage*

Disk structure

- *Disk drives are addressed as large* ***one-dimensional arrays of logical blocks,*** *where the logical block (512 bytes) is the smallest unit of transfer.*
- ***The one-dimensional array of logical blocks is mapped into the sectors of the disk sequentially.***
- ***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.***

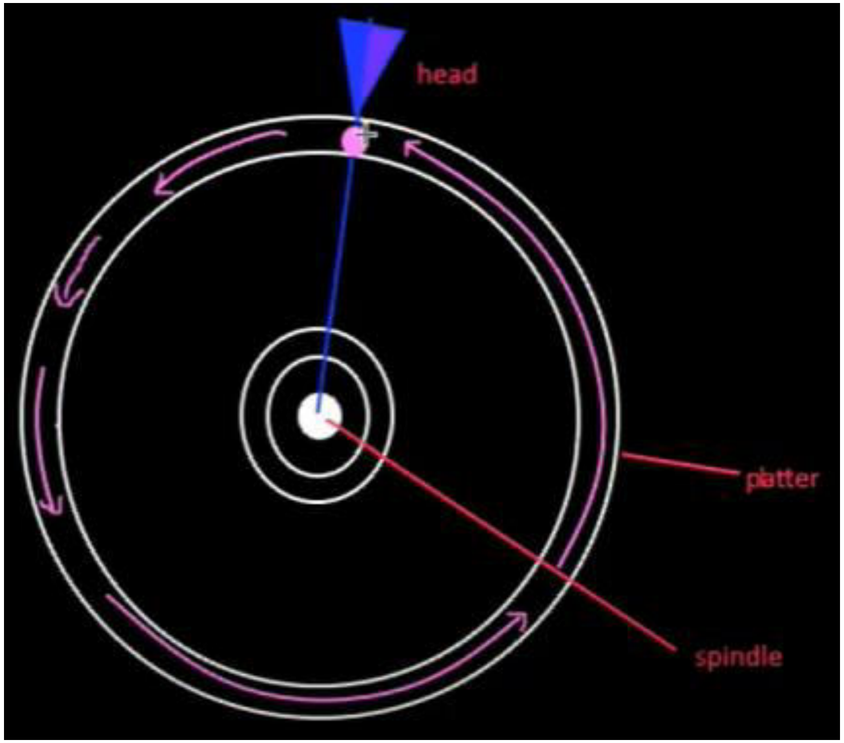
Disk Structure

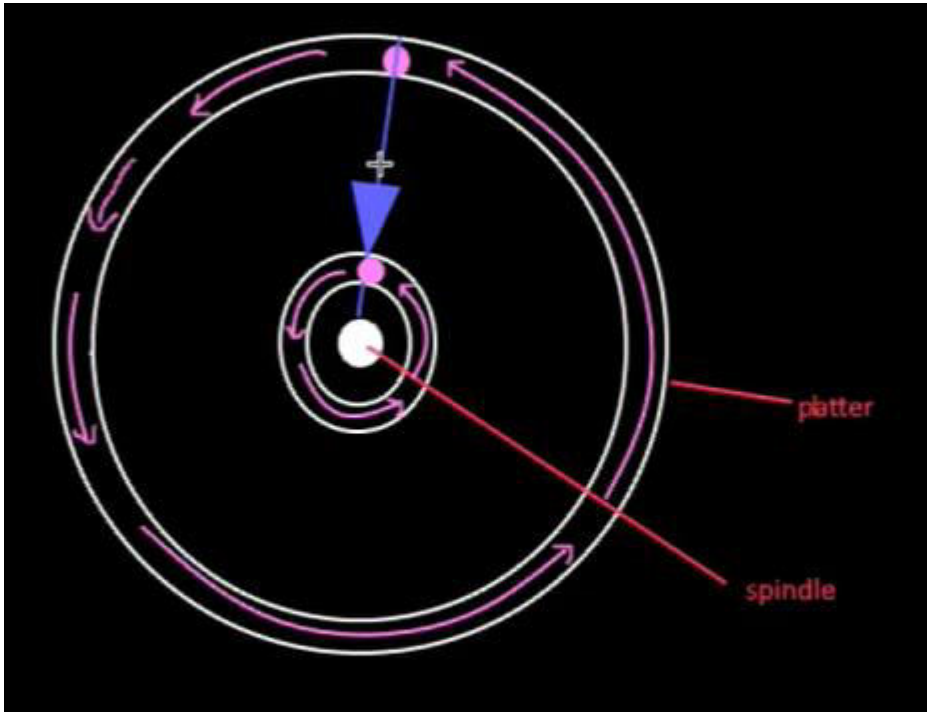
- *By use of this mapping,* ***conversion of logical to physical address*** *(consist of cylinder number, track number, sector number) should be easy.*
- *In practice it is difficult to perform this translation for two reasons. First Most disk have some defective sectors and second, the number of sectors per track is not a constant on some drive.*

Disk structure

- *Media that use* ***constant linear velocity (CLV), the density of bits per track is uniform.*** *This method is used in CD-ROM and DVD-ROM drives. Tracks in the outermost zone typically hold 40 percent more sectors than do 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, the density of bits decreases from inner tracks to outer tracks to* ***keep the data rate constant****. This method is used in hard disks and is known as* ***constant angular velocity (CAV). Number of sectors are equal on each track.***

Disk structure





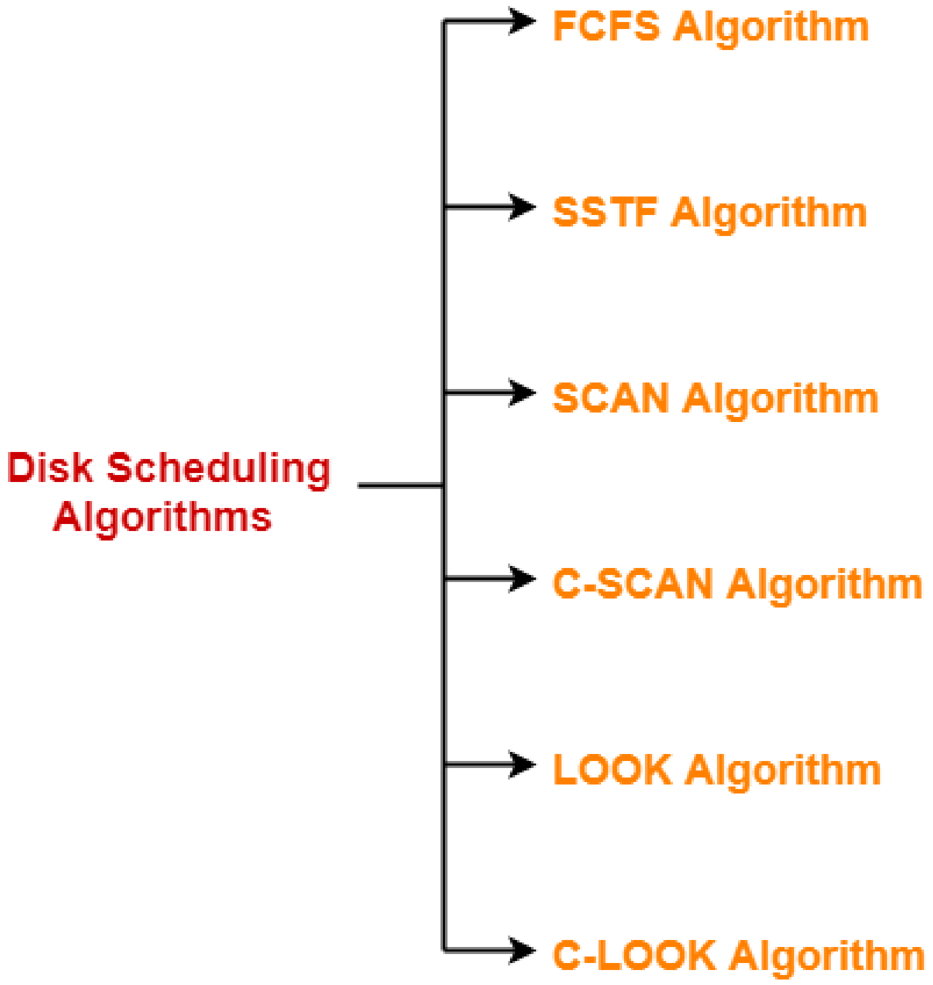
Disk Scheduling Algorithms

- *The operating system is responsible for* ***using hardware efficiently*** *— for the disk drives, this means having a* ***fast access*** *time and* ***disk bandwidth***
- ***Access time = seek time + rotational latency***
- ***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*
- ***Disk scheduling:*** *is a technique used by the operating system to schedule multiple disk I/O requests for accessing the disk.*
- *The purpose of disk scheduling algorithms is to reduce the total* ***seek time****.*

Disk Scheduling Algorithms

- *There are many sources of disk I/O request*
  - *OS*
  - *System processes*
  - *Users processes*
- *I/O request includes input/output mode, disk address, memory address, number of sectors to transfer.*
- *OS maintains* ***queue*** *of requests, per disk or device*
- *Idle disk can immediately work on I/O request, busy disk means work must queue.*

Disk Scheduling Algorithms



FCFS Algorithm

- *As the name suggests, **FCFS algorithm** entertains requests in the order they arrive in the disk queue.*
- *It is the simplest disk scheduling algorithm.*

Advantages-

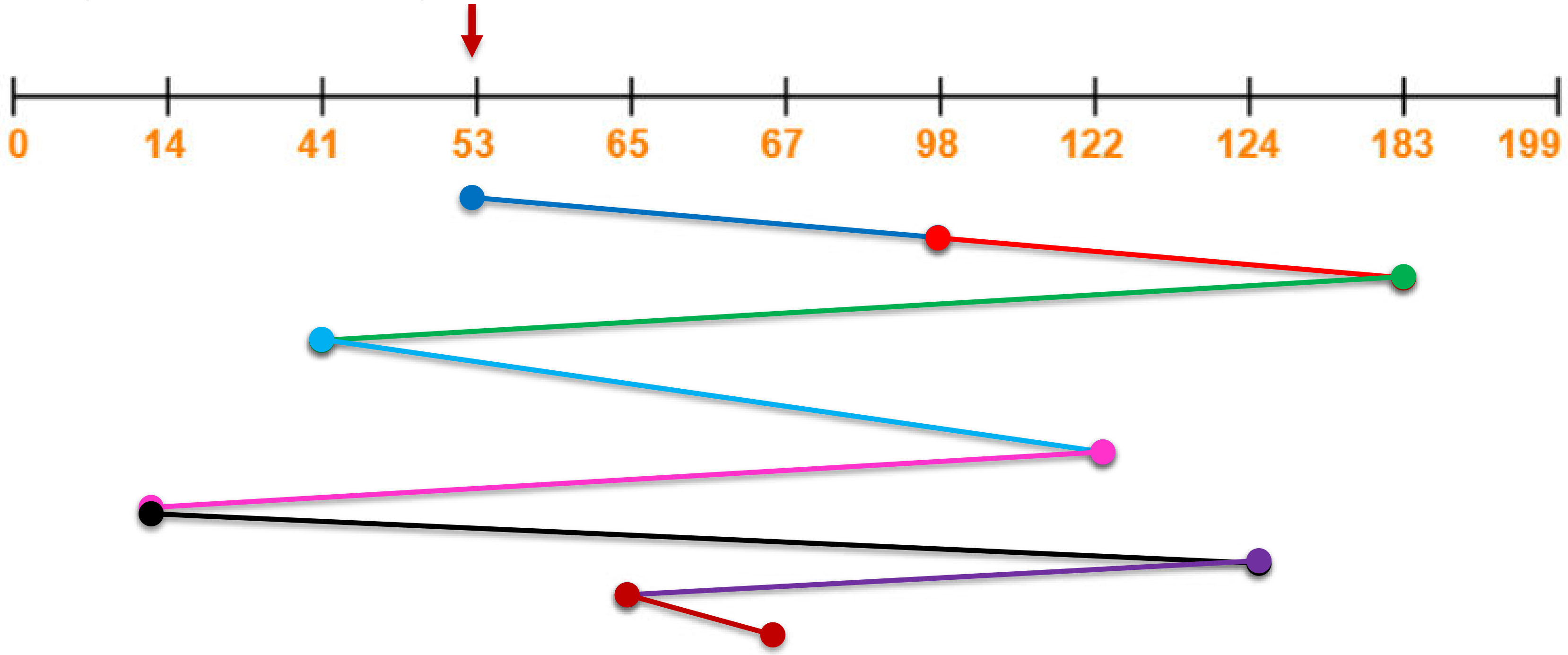
- *It is simple, easy to understand and implement.*
- *It does not cause starvation to any request.*

Disadvantages-

- *It results in an increased total seek time.*
- *It is inefficient.*

FCFS Algorithm

Consider a disk queue with requests for I/O to blocks on cylinders ***98, 183, 41, 122, 14, 124, 65, 67***. The ***FCFS*** scheduling algorithm is used. The head is initially at cylinder number ***53***. The cylinders are numbered from ***0 to 199***. Find out the total head movement (in number of cylinders) incurred while servicing these requests using ***FCFS***.



Total head movements incurred while servicing these requests

= ***(98 – 53)*** + ***(183 – 98)*** + ***(183 – 41)*** + ***(122 – 41)*** + ***(122 – 14)*** + ***(124 – 14)*** + ***(124 – 65)*** + ***(67 – 65)*** = ***45 + 85 + 142 + 81 + 108 + 110 + 59 + 2*** = ***<u>632</u>***

SSTF Disk Scheduling Algorithm

- ***SSTF*** stands for "***Shortest Seek Time First".***
- This algorithm services that request next which requires "***least number of head movements"*** from its current position regardless of the direction.
- It breaks the tie in the direction of head movement.

Advantages-

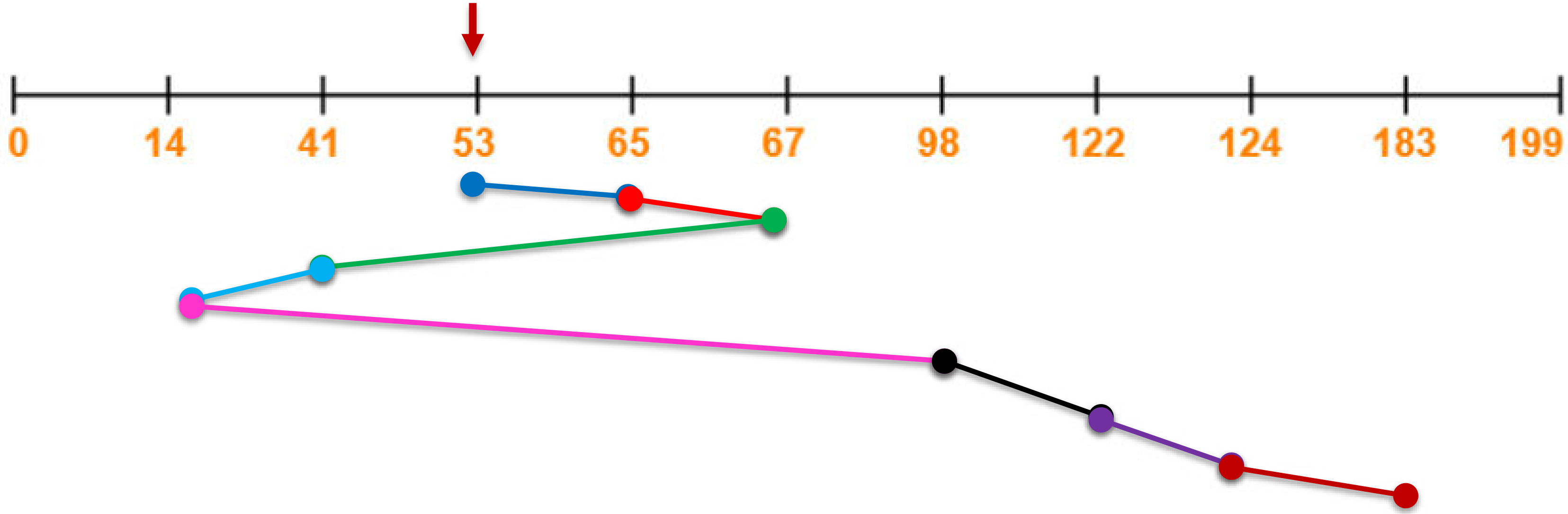
- It reduces the total seek time as compared to ***FCFS***.
- It provides increased throughput.

Disadvantages-

- There is an overhead of finding out the closest request.
- The requests which are far from the head might starve for the CPU.

SSTF Algorithm

Consider a disk queue with requests for I/O to blocks on cylinders ***98, 183, 41, 122, 14, 124, 65, 67***. The ***SSTF*** scheduling algorithm is used. The head is initially at cylinder number ***53*** moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from ***0 to 199. Find out*** total head movement (in number of cylinders) incurred.



Total head movements incurred while servicing these requests

= (65 – 53) + (67– 65) + (67 – 41) + (41-14) + (98 – 14) + (122 – 98) + (124 – 122) + (183 – 124) = 12 + 2 + 26 + 27 + 84 + 24 + 2 + 59 = <u>236</u>

SSTF Algorithm

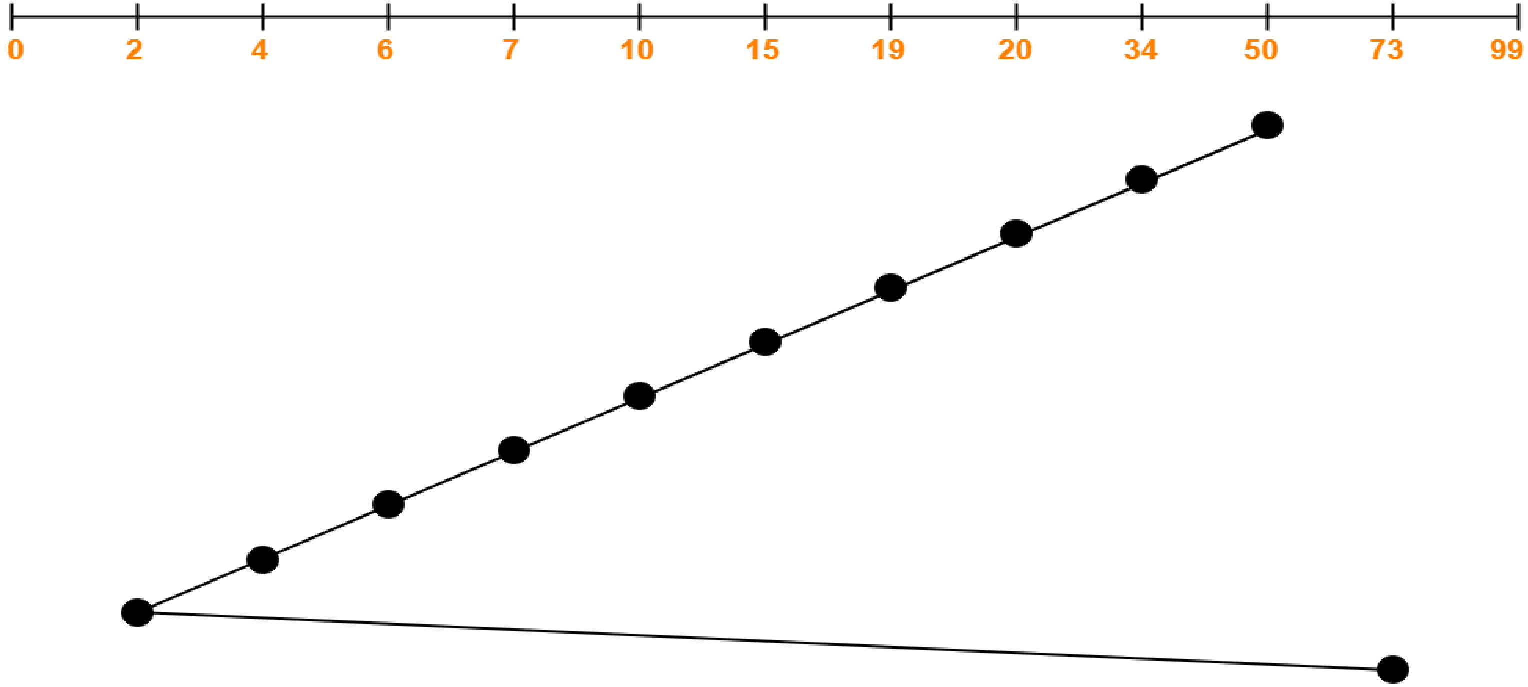
Consider a disk system with ***100*** cylinders. The requests to access the cylinders occur in following sequence-

4, 34, 10, 7, 19, 73, 2, 15, 6, 20

Assuming that the head is currently at cylinder ***50***, what is the time taken to satisfy all requests if it takes ***1 ms*** to move from one cylinder to adjacent one ***and shortest seek time first*** policy is used?

1. 95 ms
2. 119 ms
3. 233 ms
4. 276 ms

SSTF Algorithm



Total head movements incurred while servicing these requests = (50 – 34) + (34 – 20) + (20 – 19) + (19 – 15) + (15 – 10) + (10 – 7) + (7 – 6) + (6 – 4) + (4 – 2) + (73 – 2) = 16 + 14 + 1 + 4 + 5 + 3 + 1 + 2 + 2 + 71 = 119

Time taken for one head movement = 1 msec. So,

Time taken for 119 head movements

= 119 x 1 msec = 119 msec

SCAN Algorithm (Elevator Algorithm)

- As the name suggests, this algorithm scans all the cylinders of the disk ***back and forth***.
- Head starts from ***one end of the disk*** and ***move towards*** the ***other end*** servicing all the requests in between.
- After ***reaching the other end***, ***head reverses*** its direction and move towards the starting end ***servicing all the requests*** in between.

NOTE-

- SCAN Algorithm is also called as ***Elevator Algorithm*.**
- This is because its working resembles the working of an elevator.

SCAN Algorithm

Advantages-

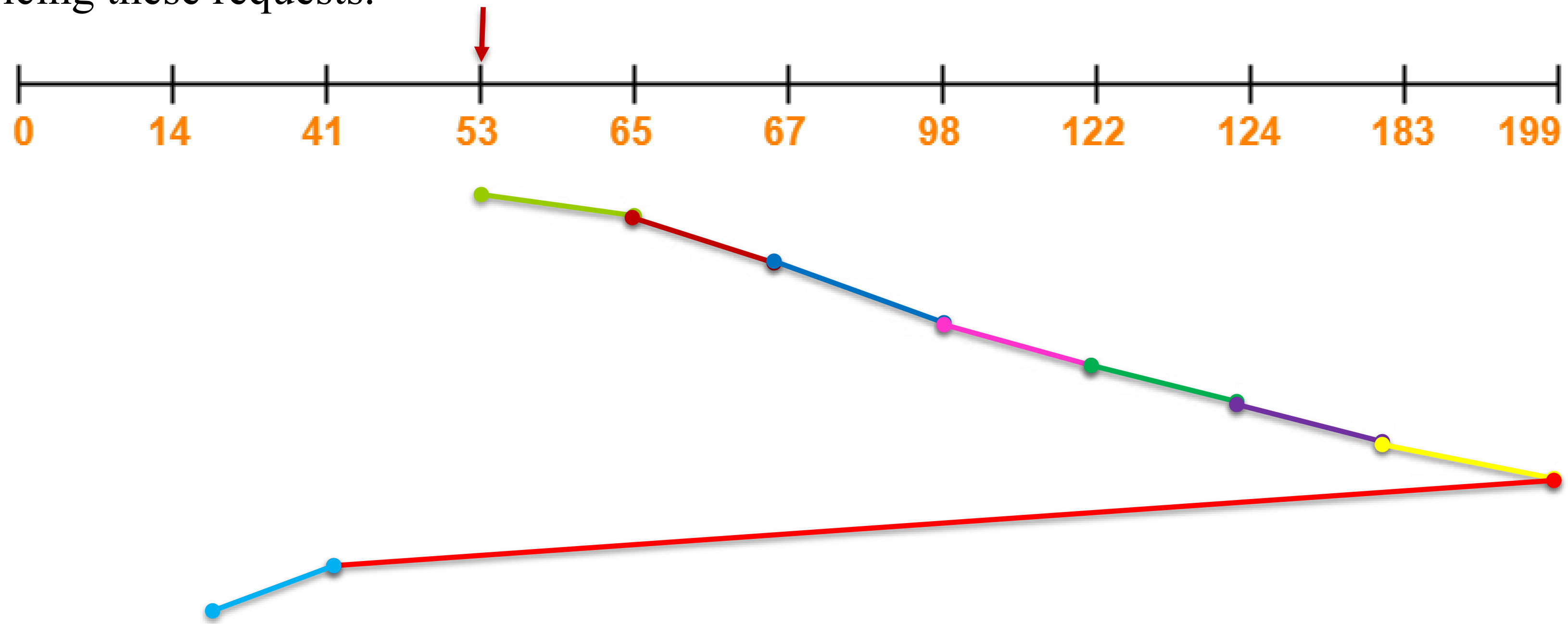
- It is simple, easy to understand and implement.
- It does not lead to starvation.

Disadvantages-

- It causes long waiting time for the cylinders just visited by the head.
- It causes the head to move till the end of the disk even if there are no requests to be serviced.

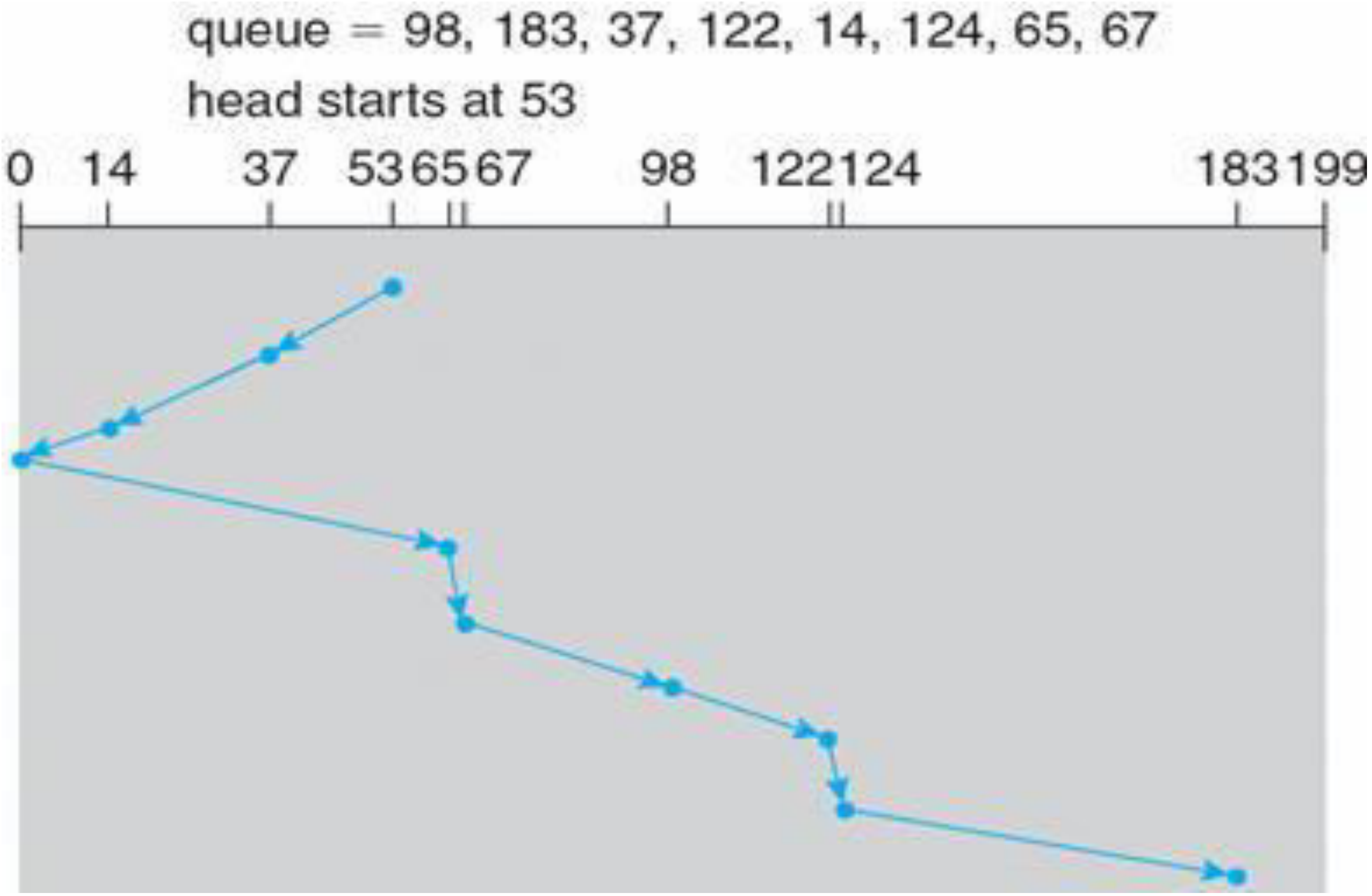
SCAN Algorithm

Consider a disk queue with requests for I/O to blocks on cylinders ***98, 183, 41, 122, 14, 124, 65, 67***. The ***SCAN*** scheduling algorithm is used. The head is initially at cylinder number 53 moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199. Find out the total number of head movement (in number of cylinders) incurred while servicing these requests.



= (65-53) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) + (199-183) + (199-41) + (41-14) = 12 + 2 + 31 + 24 + 2 + 59 + 16 + 158 + 27 = 331

SCAN Algorithm



C-SCAN Disk Scheduling

- ***Circular-SCAN*** Algorithm is a variant of the ***SCAN Algorithm***.
- Head starts from ***one end*** of the disk and move towards the ***other end*** servicing all the requests in between.
- After reaching the other end, head ***reverses*** its direction. It then returns to the ***starting end (beginning of disk)*** without servicing any request in return trip.
- The same process repeats.

C-SCAN Disk Scheduling

Advantages-

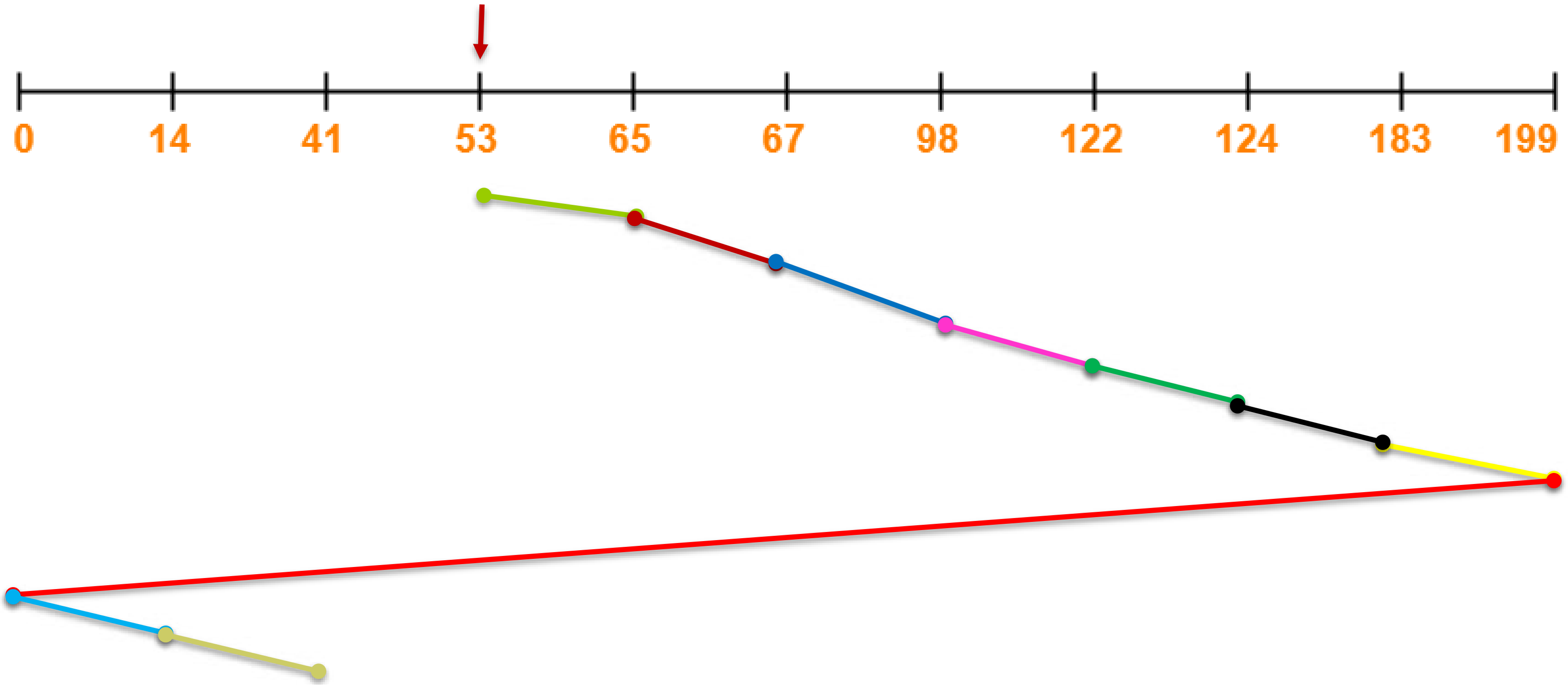
- The waiting time for the cylinders just visited by the head is reduced as compared to the ***SCAN*** Algorithm.
- It provides uniform waiting time.
- It provides better response time.

Disadvantages-

- It causes more seek movements as compared to SCAN Algorithm.
- It causes the head to move till the end of the disk even if there are no requests to be serviced.

C-SCAN Disk Scheduling

Consider a disk queue with requests for I/O to blocks on cylinders ***98, 183, 41, 122, 14, 124, 65, 67***. The C-SCAN scheduling algorithm is used. The head is initially at cylinder number ***53*** moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199. Find out the total head movement (in number of cylinders) incurred.



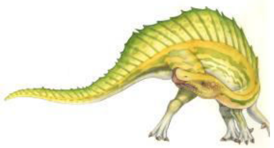
= (65 – 53) + (67 – 65) + (98 – 67) + (122 – 98) + (124 – 122) + (183 – 124) + (199 – 183) + (199 – 0) + (14 – 0) + (41 – 14) = 12 + 2 + 31 + 24 + 2 + 59 + 16 + 199 + 14 + 27 = 386

LOOK Disk Scheduling

- ***LOOK*** Algorithm is an improved version of the *SCAN Algorithm*.
- ***Head*** starts from the ***first request*** at one end of the disk and moves towards the ***last request*** at the other end servicing all the requests in between.
- After reaching the ***last request*** at the other end, head ***reverses*** its direction.
- It then returns to the ***first*** request at the ***starting end*** servicing all the requests in between.
- The same process repeats

Advantages-

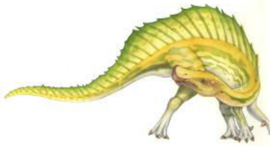
- It does not causes the head to move till the ends of the disk when there are no requests to be serviced.
- It provides better performance as compared to SCAN Algorithm.
- It does not lead to starvation.



LOOK Disk Scheduling

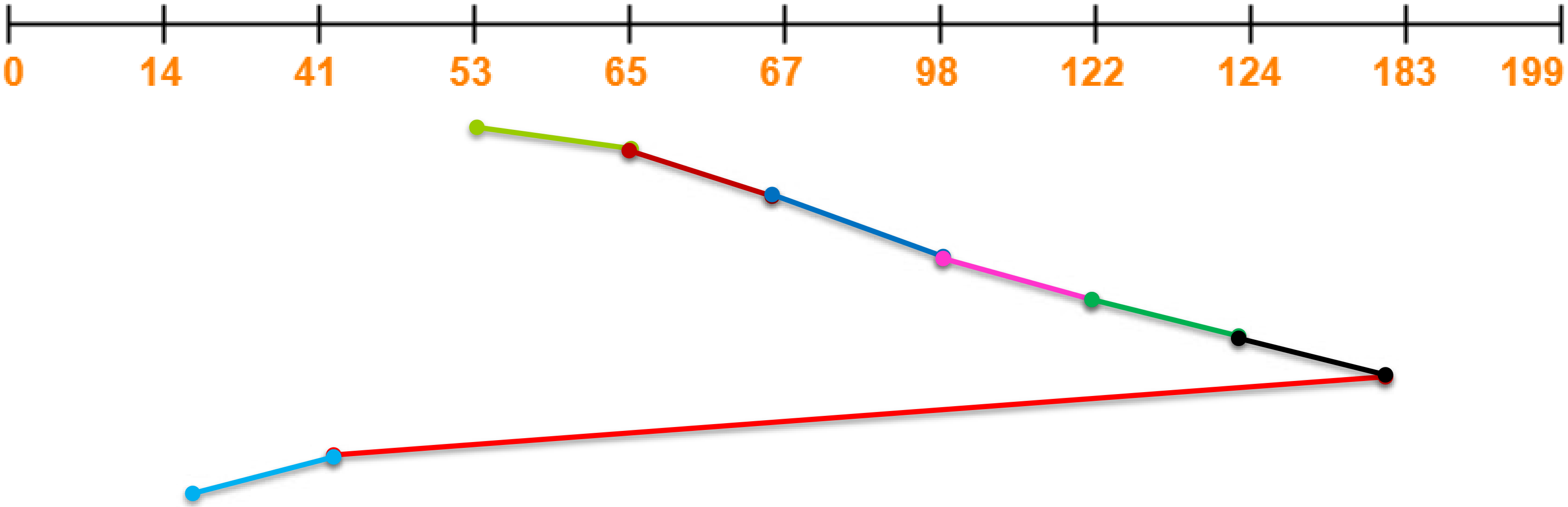
Disadvantages-

- There is an overhead of finding the end requests.
- It causes long waiting time for the cylinders just visited by the head.



LOOK Disk Scheduling

Consider a disk queue with requests for I/O to blocks on cylinders ***98, 183, 41, 122, 14, 124, 65, 67.*** The LOOK scheduling algorithm is used. The head is initially at cylinder number 53 ***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



= (65 – 53) + (67 – 65) + (98 – 67) + (122 – 98) + (124 – 122) + (183 – 124) + (183 – 41) + (41 – 14)

= 12 + 2 + 31 + 24 + 2 + 59 + 142 + 27

= ***299***

C-LOOK Disk Scheduling

- ***Circular-LOOK*** Algorithm is a variant of the ***LOOK Algorithm***.
- ***Head*** starts from the first request at one end of the disk and moves towards the ***last request*** at the other end servicing all the requests in between.
- After reaching the last request at the other end, head reverses its direction. It then returns to the ***first request*** at the starting end without servicing any request in between.
- The same process repeats.

C-LOOK Disk Scheduling

Advantages-

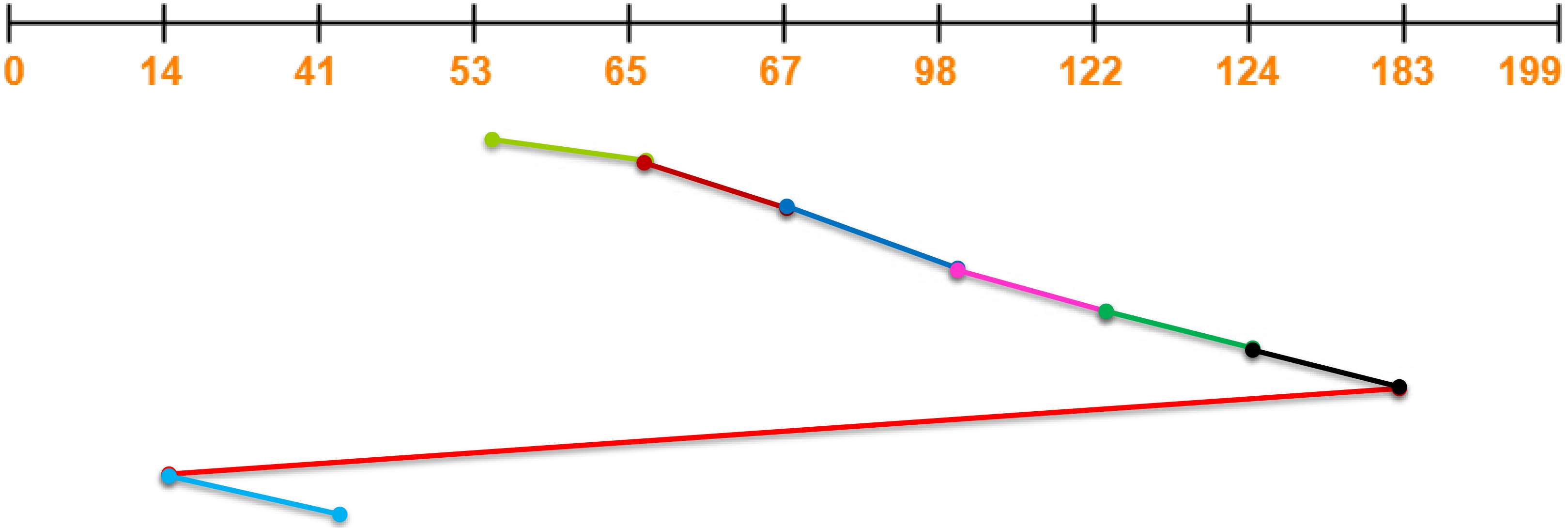
- It does not causes the head to move till the ends of the disk when there are no requests to be serviced.
- It reduces the waiting time for the cylinders just visited by the head.
- It provides better performance as compared to LOOK Algorithm.
- It does not lead to starvation.

Disadvantages-

- There is an overhead of finding the end requests.

C-LOOK Disk Scheduling

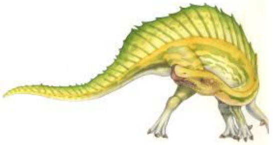
Consider a disk queue with requests for I/O to blocks on cylinders ***98, 183, 41, 122, 14, 124, 65, 67***. The C-LOOK scheduling algorithm is used. The head is initially at cylinder number 53 moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred.



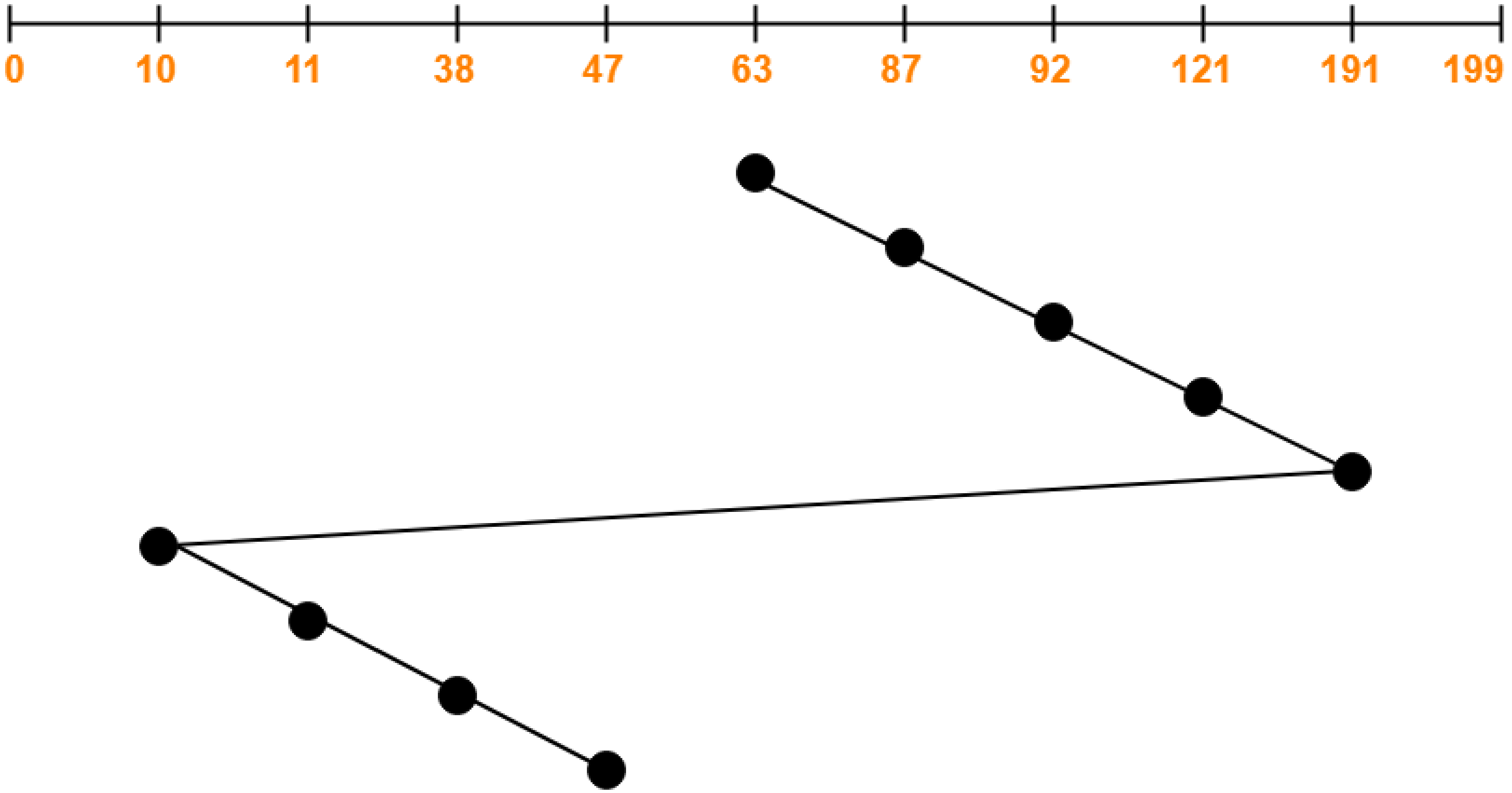
= (65 – 53) + (67 – 65) + (98 – 67) + (122 – 98) + (124 – 122) + (183 – 124) + (183 – 14) + (41 – 14)
= 12 + 2 + 31 + 24 + 2 + 59 + 169 + 27
= 326

C-LOOK Disk Scheduling

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



= (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

Selection of a Disk-Scheduling Algorithm

- With very low loads all algorithms are equal, since there will normally only be one request to process at a time.
- For slightly larger loads, SSTF offers better performance than FCFS, but may lead to starvation when loads become heavy enough.
- For busier systems, SCAN and LOOK algorithms eliminate starvation problems.

Disk Management - Disk Formatting

- *A new 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 known as **Low level formatting** or **physical formatting**.*
- *This **Low level formatting** fill each sector with a special data structure: **header – data area** (Usually 512 bytes of data but can be selectable) **– trailer.***
- *Header and Trailer contains information such as a **sector number** and an **error-correcting code (ECC).***
- *When the controller **writes** a sector of data – ECC get updated with a value calculated from all the bytes in the data area.*

Disk Formatting

- *When the sector is **read**, ECC is recalculated and is compared with the stored value and verify the data is correct (stored and recalculated ECC is same) or corrupted.*
- *ECC calculation is performed with every disk read or write, and if damage is detected but the data is recoverable, then a **soft error** has occurred.*
- *Soft errors are generally handled by the on-board disk controller, and never seen by the OS.*

Disk Formatting

- ***To use a disk to hold files***, *the operating system needs to record its* ***own data structures*** *on the disk. It does so in two steps:*

  1. ***Logical formatting*** *or "making a file system" : In this step OS stores the initial file system data structure onto the disk, which maps to the free and allocated space (FAT, inode) and an initial empty directory.*
  2. ***Partition:*** *partition the disk into one or more groups of cylinders, each partition is treated as a* ***separate logical disk****.*

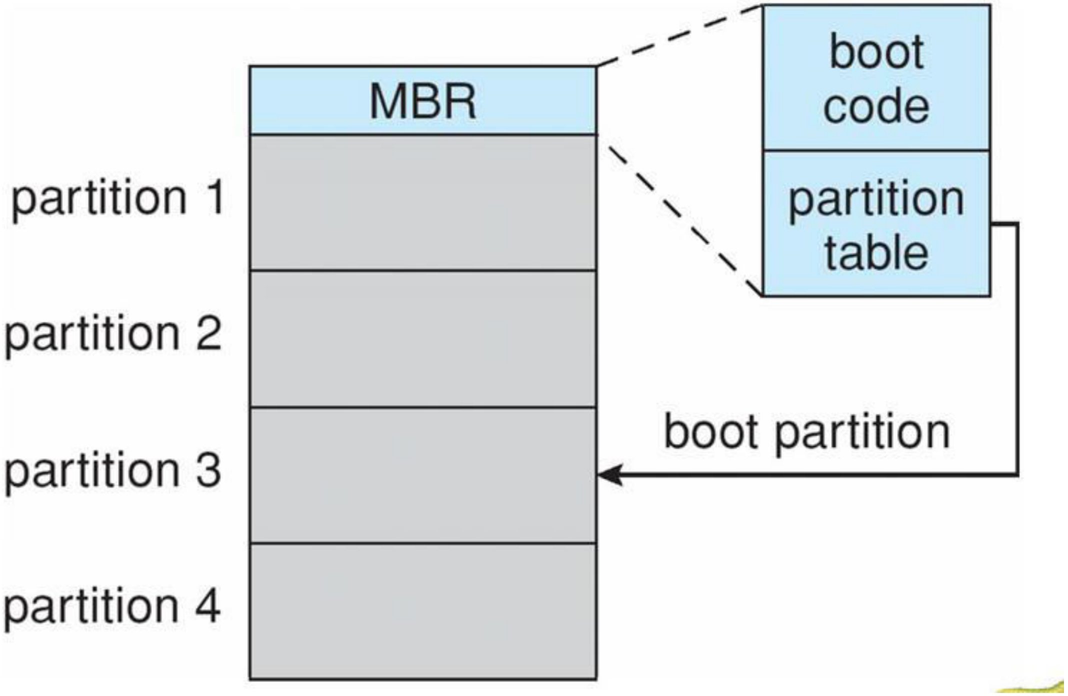
Boot Block

- ***Bootstrap program*** *initializes all aspect of system. –* ***Initialize CPU registers, device controllers, main memory –Start OS***
- *To do its job, 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.*
- *In PC,* ***two-step*** *approaches –A* ***tiny bootstrap program*** *is stored in ROM, ROM do not need initialization and placed at fixed location which processor can start executing when powered up. Since ROM is read only it can not be infected by computer* ***virus****.*
- *ROM is faster as it is directly placed at motherboard of system.*

Boot Block

- *Changing the bootstrap code basically requires changes in the ROM hardware chips.*
- *Most system nowadays has the tiny bootstrap loader program in the ROM whose only job is to bring the full bootstrap program from the disk.*
- *The* ***full bootstrap*** *program is stored in the* ***boot blocks*** *at fixed location over the disk.*

Booting from disk in windows



The Master Boot Record (MBR) is the information in the first sector of a hard disk or a removable drive. It identifies how and where the system's operating system (OS) is located in order to be booted (loaded) into the computer's main storage or random access memory (RAM).

Bad Blocks (Defective Blocks)

Bad Block *is an area of storing media that is* ***no longer reliable*** *for the storage of data because it is completely damaged or corrupted.*

Simple disk (IDE Controllers): *Bad blocks are handled* ***manually.***

- *One strategy is to* ***scan the disk*** *to find bad blocks, any bad block that are discovered are flagged as unusable, so that file system does not allocate them.*
- *If blocks go bad during operation, a* ***special program*** *(such as badblocks command in linux) must be run manually to search and lock the bad blocks.*

Bad Blocks (Defective Blocks)

More sophisticated disks are smarter about bad-block recovery (SCSI)

- Controller maintains a list of bad blocks on the disk, initialized at the time of low level formatting.
- Low-level formatting will set aside spare sectors not visible to OS.

- ***Sector sparing (or forwarding):***

Controller can be told to replaces each bad sector logically with one of the spare sectors. A typical bad-sector transaction is as follows –

- Suppose Operating system wants to read logical block 80.
- Now, the controller is going to calculate ECC and suppose it found the block as bad. It reports to operating system that the requested block is bad.

Bad Blocks (Defective Blocks)

- Whenever, next time the system is rebooted, a special command is used and it will tell the controller that this sector is to be replaced with the spare sector.
- In future, whenever there is a request for the block 80, the request is translated to replacement sector's address by the controller.

Sol: Each cylinder has a few spare sectors

For this reason, most disks are formatted to provide a few spare sectors in each cylinder and spare cylinder as well. Whenever the bad block is going to remap, the controller will use spare sector from the same cylinder, if possible; otherwise spare cylinder is also present.

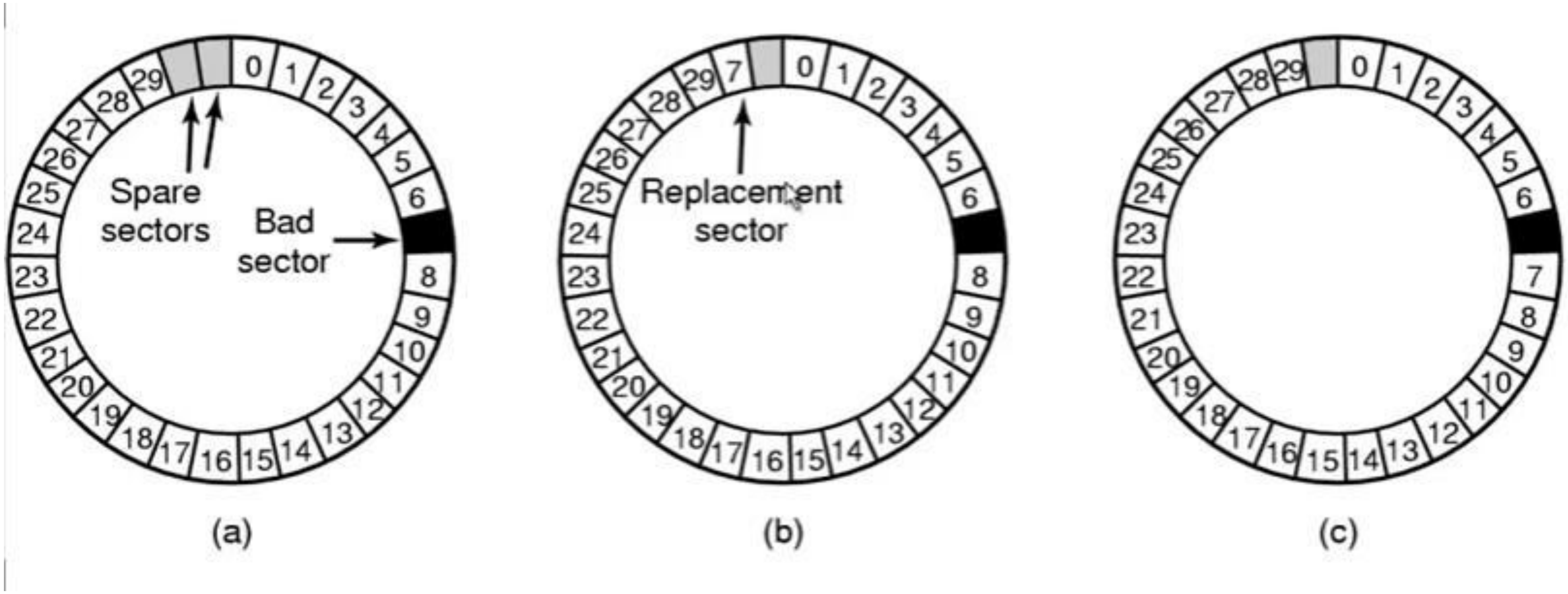
Bad Blocks (Defective Blocks)

- Another technique: ***sector slipping***
  - Suppose that logical block 16 becomes defective and the first available spare sector follows sector 200.
  - Sector slipping then starts remapping. All the sectors from 16 to 200, moving them all down one spot.
  - That is, sector 200 is copied into the spare, then sector 199 into 200, then 198 into 199, and so on, until sector 17 is copied into sector 18.
  - In this way slipping the sectors frees up the space of sector 17 so that sector 16 can be mapped to it.

Bad Blocks (Defective Blocks)

- The replacement of bad block is not totally automatic, because data in the bad block are usually lost.
- A process is trigger by the ***soft errors*** in which a copy of the block data is made and the block is **spared or slipped**.
- ***Hard error*** which is unrecoverable will lost all its data. Whatever file was using that block must be repaired and that requires manual intervention.

Bad Blocks (Defective Blocks)



(a)

(b)

(c)

RAID Structure

- ***RAID*** *stands for "**Redundant Array of Independent Disks**. "*

  Redundant Arrays of Inexpensive Disks.

- ***RAID*** *is a technique which makes use of a combination of **multiple disks** instead of using a single disk:*
  - *In order to provide **fault tolerance**,*
  - *To improve overall **performance**,*
  - *To maintain **data redundancy***
  - *To increase **storage capacity** in a system*

RAID Improve Reliability via Redundancy

Disk failure will result in loss of significant amount of data.

- ***Solutions to reliability: Redundancy***
  - ***Mirroring: Duplicate every disk*** *-> a logical disk will consist of two physical disk and every write is carried out on both disks.*
  - *The same data was copied onto multiple disks, then the data would not be lost unless **both** (or all) copies of the data were damaged or the second disk would have to go bad before the first disk was repaired, which brings the **Mean Time To Repair** into play.*
  - ***Parity***

RAID Improve Reliability via Redundancy

- ***Mean time to data loss***, *depends on*
  - ***Mean time to failure of*** *individual disk*.
  - ***Mean time to repair-*** *the time to replace a failed disk and to restore the data.*

$$MTTDL = (MTTF ^{} 2) / ((\#Disks) * (\#Disks - 1) * MTTR)$$

For example if two disks were involved, each with a MTTF of 100,000 hours and a MTTR of 10 hours, then the ***Mean Time to Data Loss*** *would be*

$$= 100{,}000^2 / (2*10)$$

$= 500 * 10^6$ *hours, or 57,000 years!*

Improvement in Performance via Parallelism

- *With respect to **reads.** Since every block of data is duplicated on multiple disks (mirroring), **read** operations can be satisfied from **any available disk**,*
- *Multiple disks can be reading different data blocks simultaneously in parallel. The transfer rate of each read is same as a single disk system.*

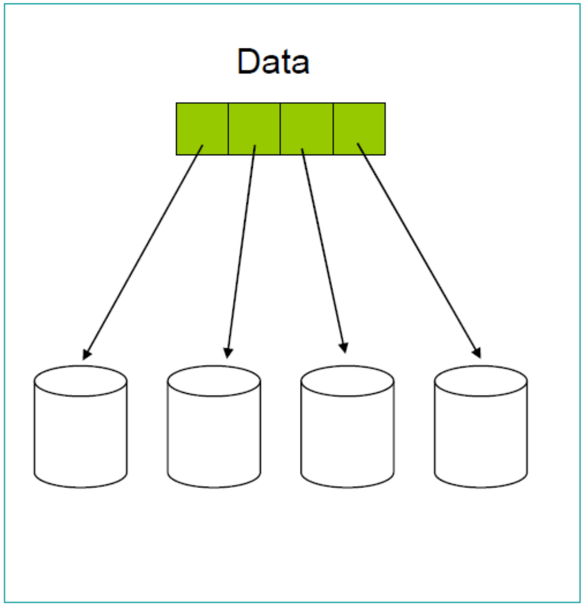
Improvement in Performance via Parallelism

- ***Striping:*** *Another way of improving disk access time is with* ***striping****, which basically means spreading* ***data*** *out across multiple disks that can be* ***accessed simultaneously****.*

- ***Data striping***
  - ***Bit-level striping:*** *With* ***bit-level striping*** *the bits of each byte are striped across multiple disks. For example if 8 disks were involved, we write bit i of each byte to disk i, then each 8-bit byte would be read in parallel by 8 heads on separate disks.*

Improvement in Performance via Parallelism

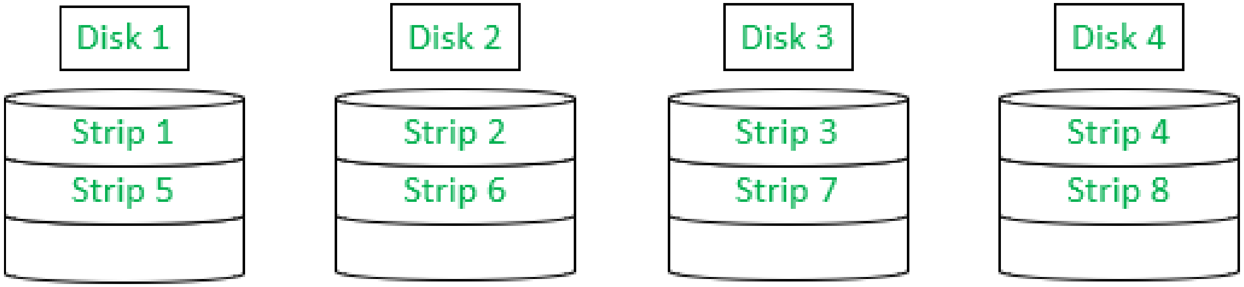
- ***Block-level striping:*** *spreads a file system across multiple disks on a block-by-block basis, so if block N were located on disk 0, then block N + 1 would be on disk 1, and so on, with n disks, block i goes to disk (i mod n)+1*



RAID Levels

- Mirroring provides reliability but is expensive; S
- Striping improves data transfer rate, but does not improve reliability.
- Accordingly there are a number of different schemes that combine the principals of mirroring and striping in different ways, in order to balance reliability versus performance versus cost. These are described by different ***RAID levels***, as follows:

Raid Level 0 (Striping)- This configuration has striping, but no redundancy of data. It offers the best performance, but it does not provide fault tolerance. If one drive fails then all data in the array is lost

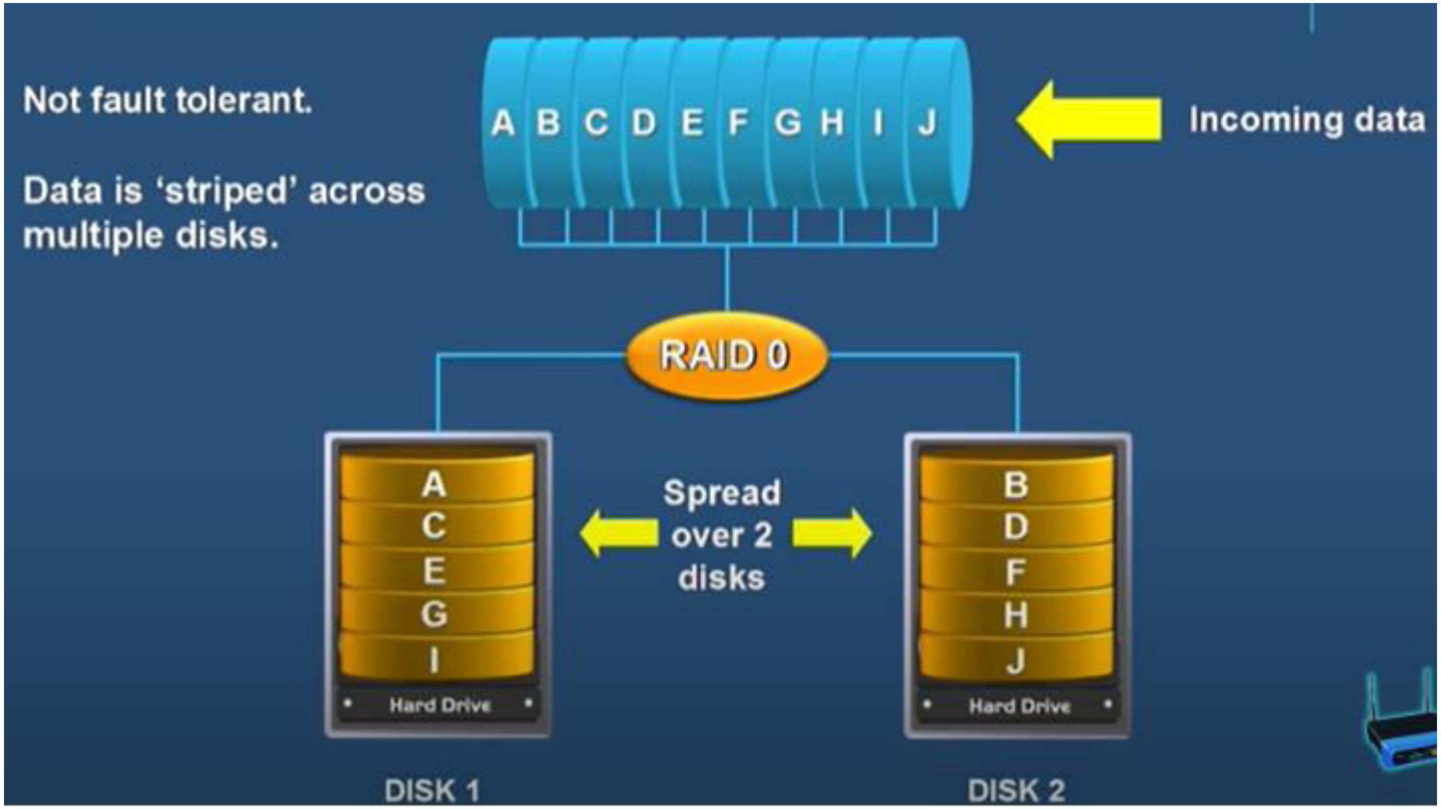


RAID Levels

- ***Mirroring provides reliability*** *but is expensive;*
- ***Striping improves data transfer rate, but does not improve reliability.***
- *Accordingly there are a number of different schemes that combine the principals of mirroring and striping in different ways, in order to balance reliability versus performance versus cost. These are described by different* ***RAID levels****, as follows:*

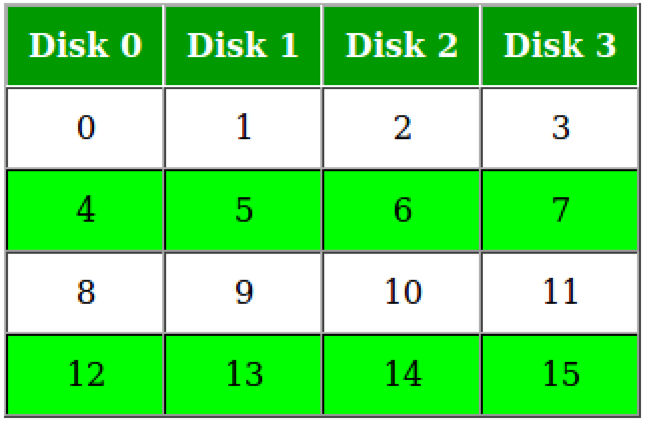
RAID-0 (Striping)

Blocks are "stripped" across disks.



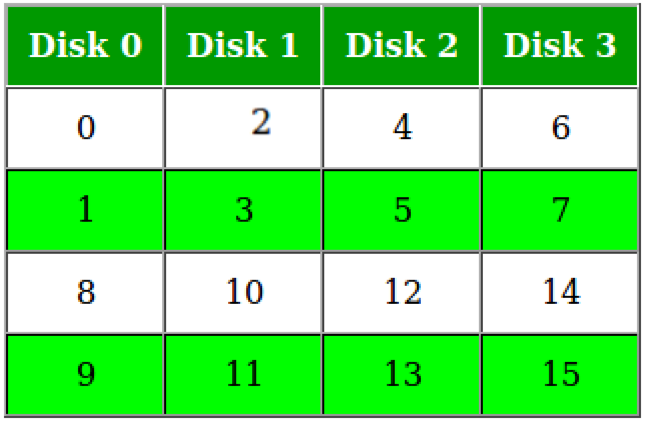
RAID-0 (Striping)

Blocks are "striped" across disks



Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Instead of placing just one block into a disk at a time, we can work with two (or more) blocks placed into a disk before moving on to the next one.



Disk 0	Disk 1	Disk 2	Disk 3
0	2	4	6
1	3	5	7
8	10	12	14
9	11	13	15

RAID-0 (Striping)

Evaluation:

- *Reliability: 0*

  There is no duplication of data. Hence, a block once lost cannot be recovered.

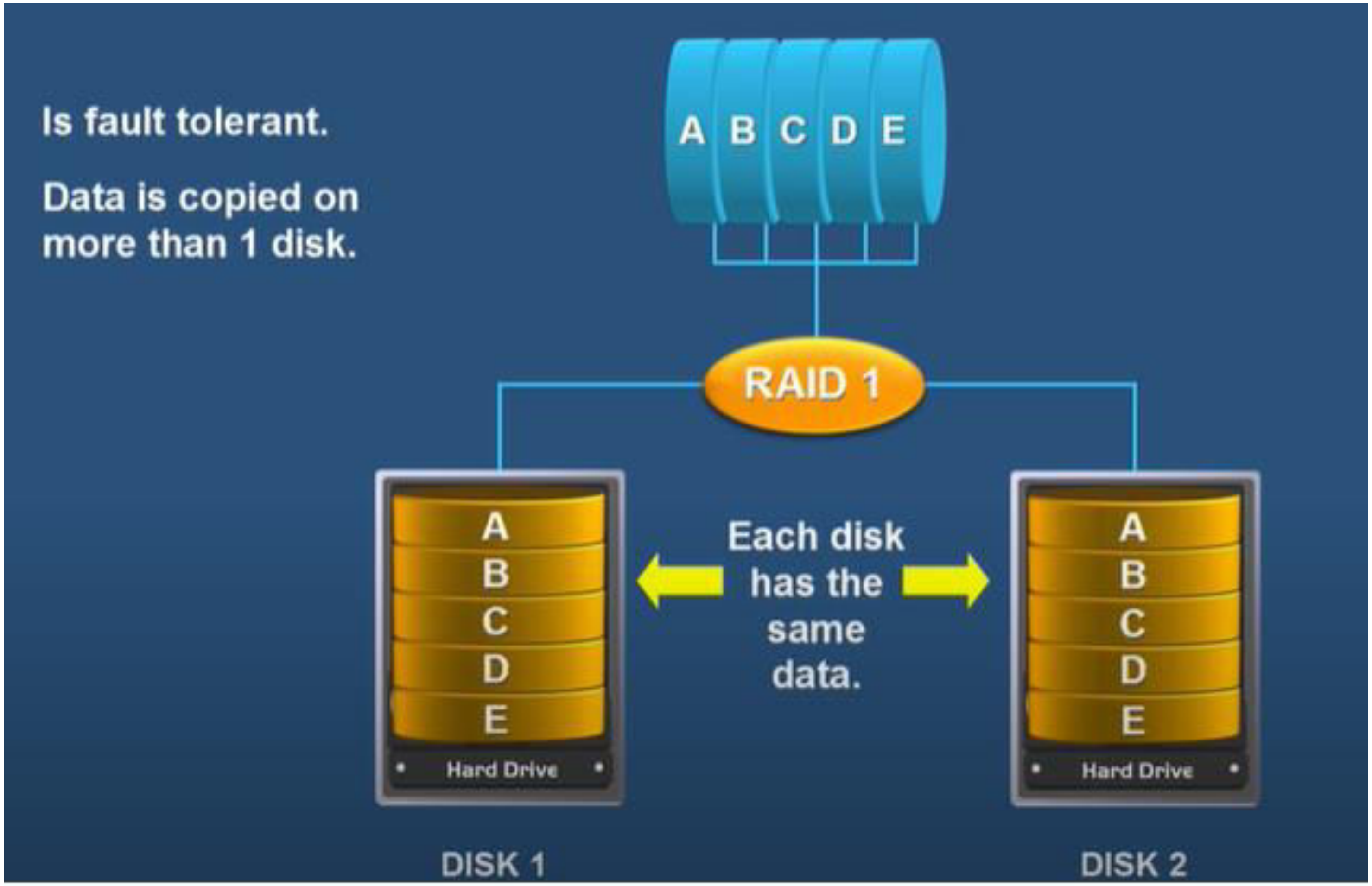
- *Capacity: N*B*

  The entire space is being used to store data. Since there is no duplication, N disks each having B blocks are fully utilized.

RAID-1 (Mirroring)

- ***RAID-1*** *also known as* ***disk mirroring***.
- *More than one copy of each block is stored in a separate disk. Thus, every block has two (or more) copies, lying on different disks.*
- *There is no striping.*
- *RAID 1 provides twice the read transaction rate of single disks and the same write transaction rate as single disks.*

RAID-1 (Mirroring & Duplexing)



Is fault tolerant.

Data is copied on more than 1 disk.

Each disk has the same data.

RAID-1 (Mirroring)

RAID 0 was unable to tolerate any disk failure. But RAID 1 is capable of reliability.

Evaluation: *Assume a RAID system with mirroring level 2.*

- ***Reliability:***

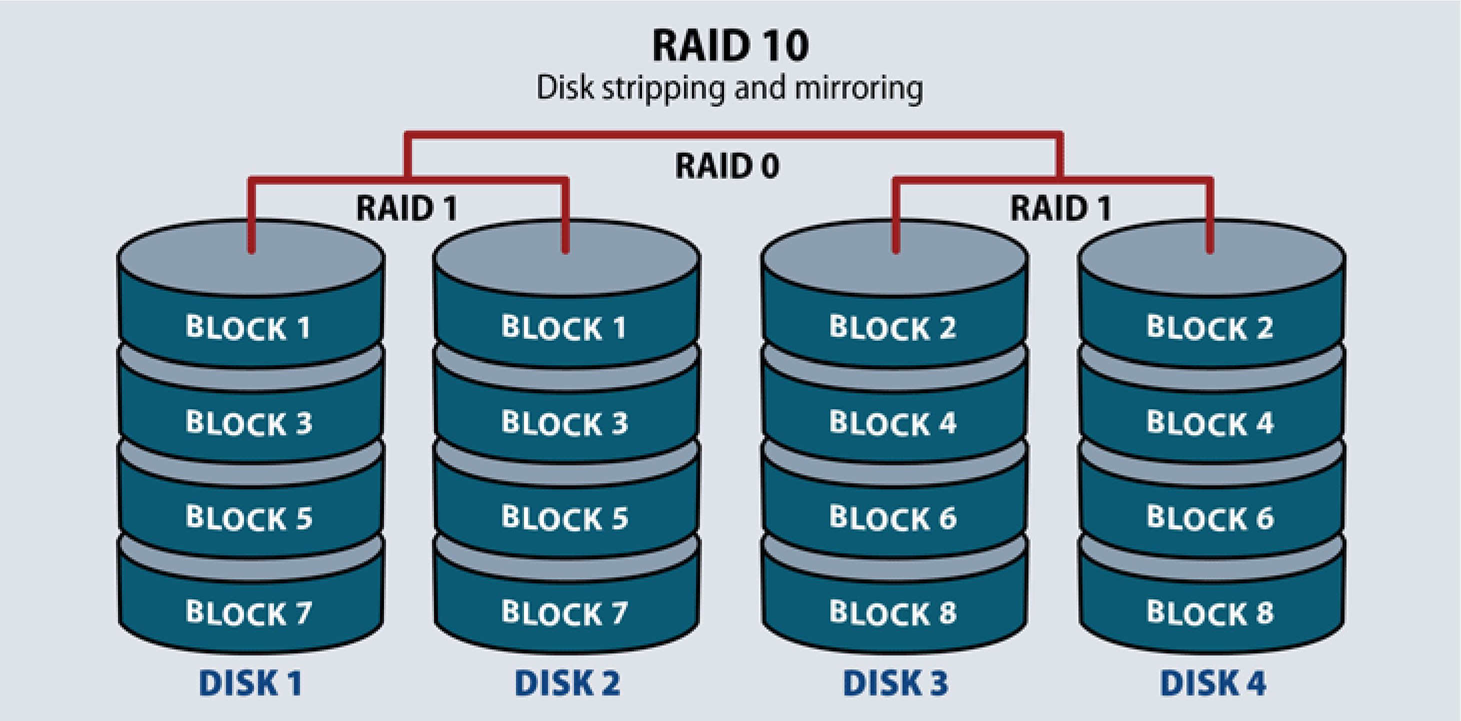
  N/2 disk failures can be handled.

- ***Capacity: N*B/2***

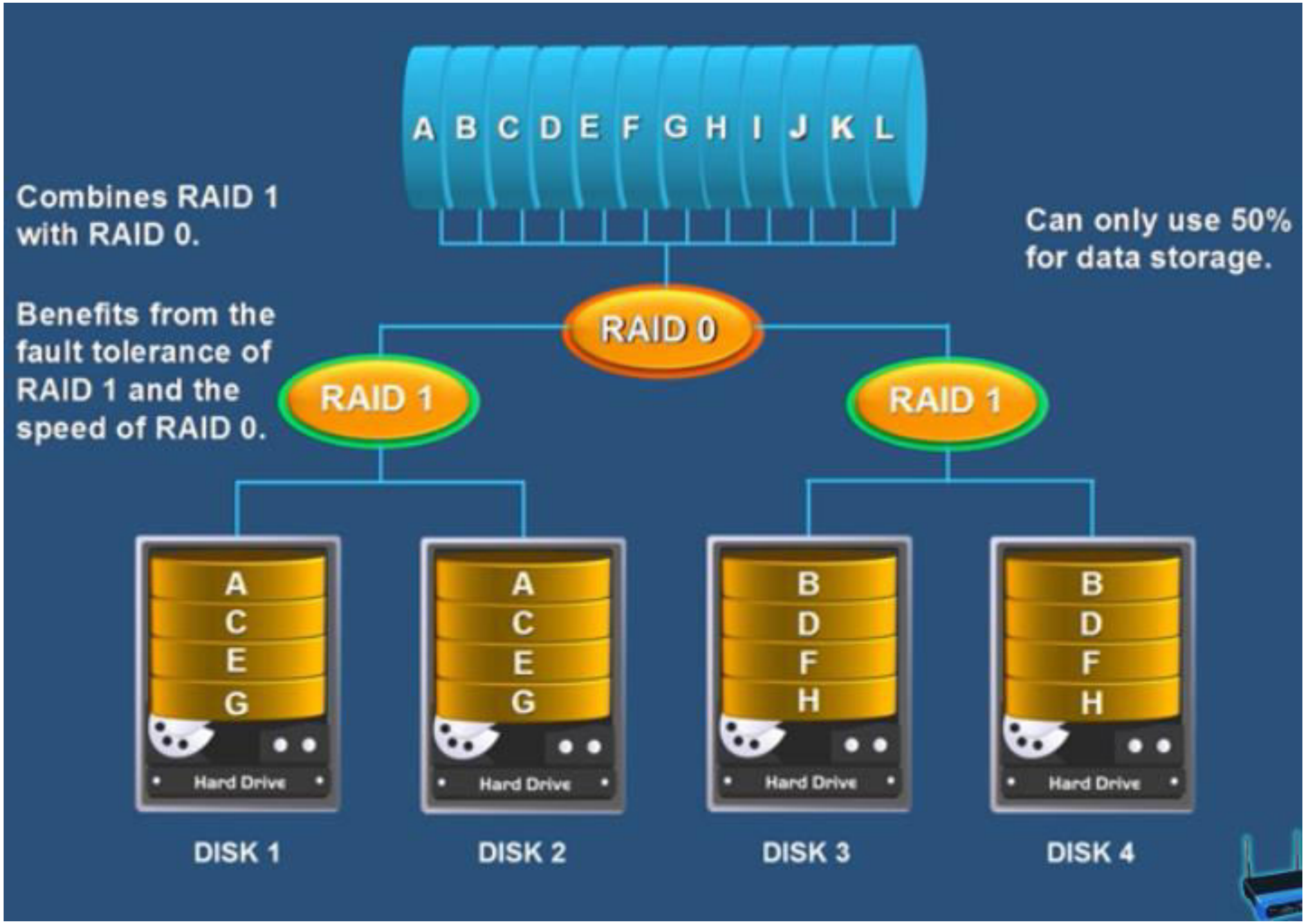
  Only half the space is being used to store data. The other half is just a mirror to the already stored data.

RAID-10 (0 + 1 and 1+ 0)

RAID 10 requires a minimum of four disks in the array. It stripes across disks for higher performance, and mirrors for redundancy. In a four-drive array, the system stripes data to two of the disks. The remaining two disks mirror the striped disks, each one storing half of the data.



RAID-10 (0 + 1 and 1+ 0)

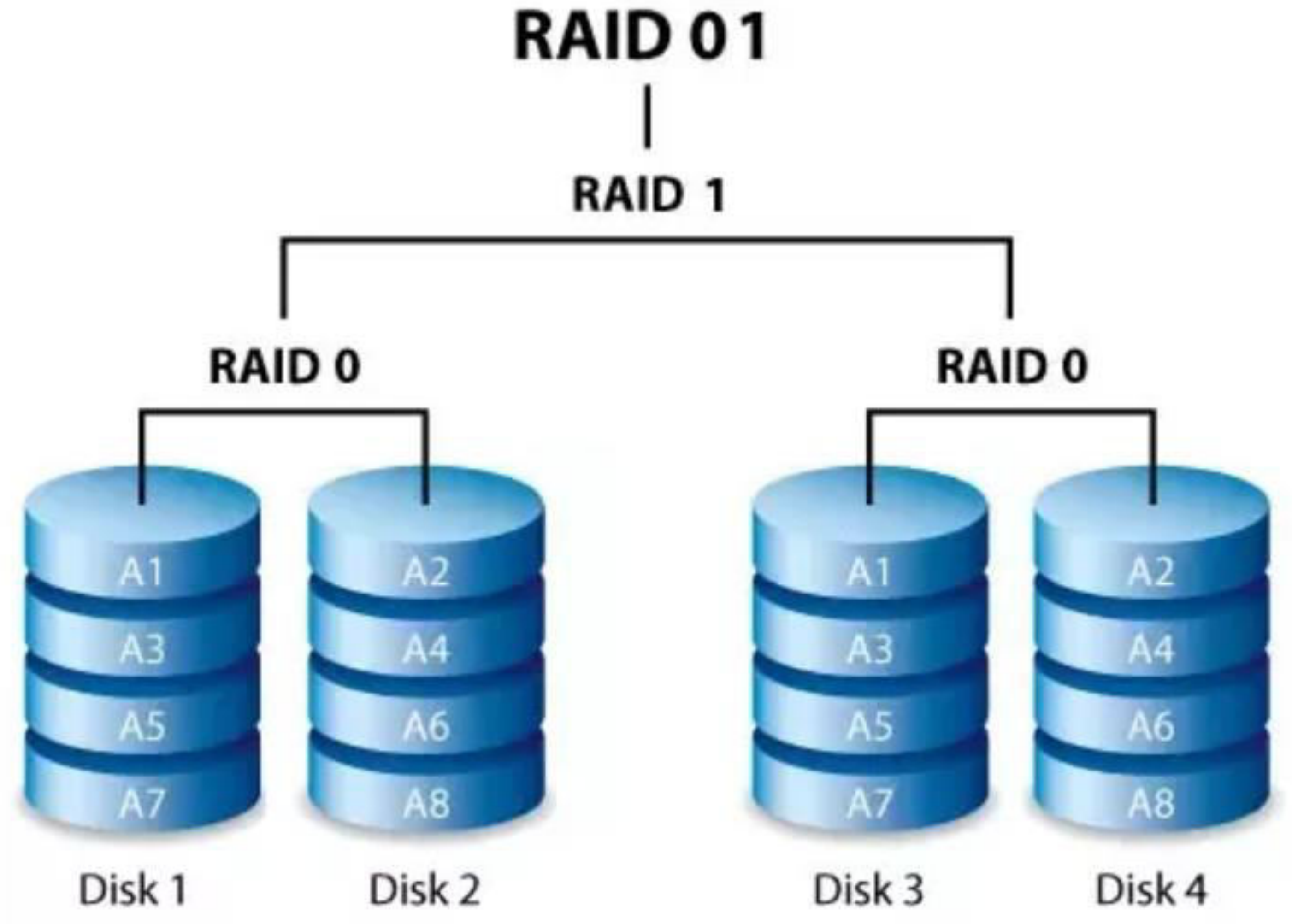


Combines RAID 1 with RAID 0.

Benefits from the fault tolerance of RAID 1 and the speed of RAID 0.

Can only use 50% for data storage.

RAID-10 (0 + 1 and 1+ 0)



RAID-2 (Memory Style Error-Correcting-Code (ECC) Organization)

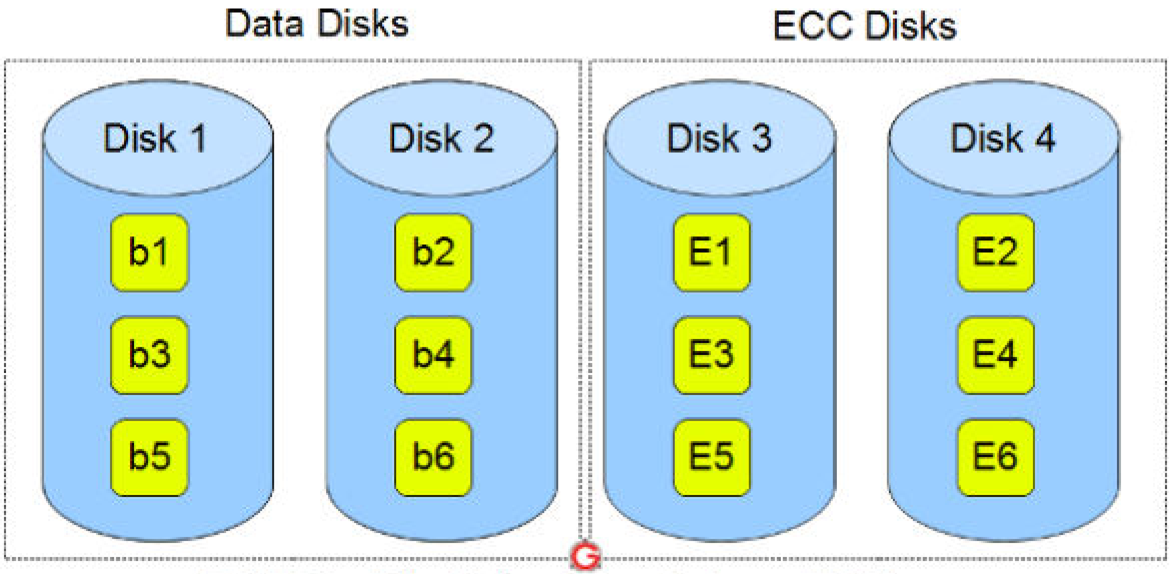
- *This level uses **bit-level data stripping** rather than block level. To be able to use RAID 2 make sure the disk selected has **no self disk error checking** mechanism. This level uses external **Hamming code** for error detection.*
- *Error-correcting schemes (ECC) store two or more extra bits and it is used for reconstruction of the data if a single bit is damaged.*
- ***Hamming Code :** It is an algorithm used for error detection and correction when the data is being transferred. It adds extra, redundant bits to the data.*

RAID-2 (Memory style error-correcting-code (ECC) organization)

- You need ***two groups*** of the disks. One group of disks are used to write the data, another group is used to write the ***Hamming error correction code (ECC)*** in redundancy disk.
- When data is ***written*** to the disks, it calculates the ECC code for the data and ***stripes*** the data bits to the ***data-disks***, and writes the ECC code to the ***redundancy disks.***
- When data is ***read*** from the disks, it also reads the corresponding ECC code from the redundancy disks, and checks whether the data is consistent. If required, it makes appropriate corrections.

RAID - 2 (Memory style error-correcting-code (ECC) organization)

- *This is not used anymore. This is* ***expensive*** *and implementing it in a RAID controller is* ***complex****, and* ***ECC*** *is* ***redundant*** *now-a-days, as the hard disk themselves can do this.*
- *Actual number of redundant disks will depend on* ***correction model***

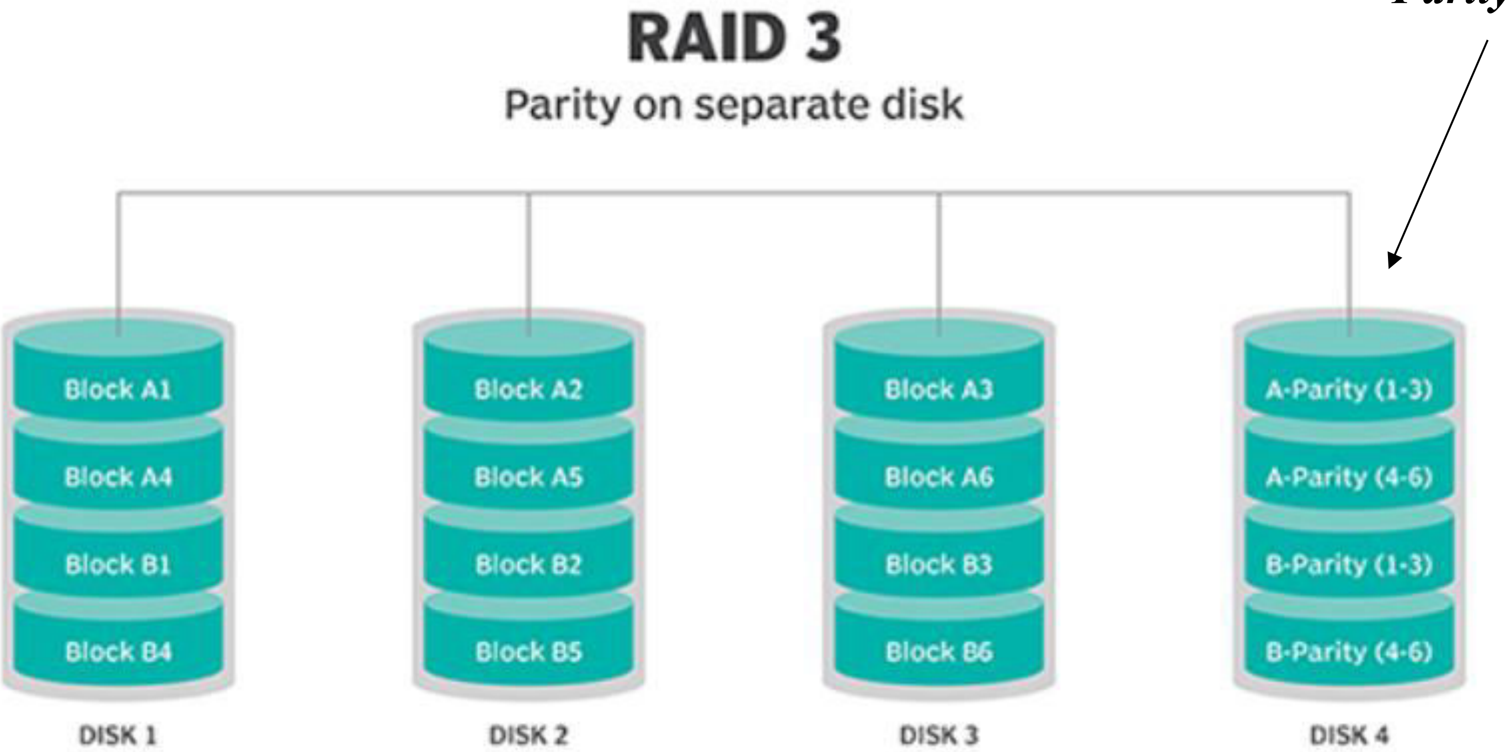


RAID 2 – Bits Striped. (and stores ECC)

RAID-3 (Bit - Interleaved Parity)

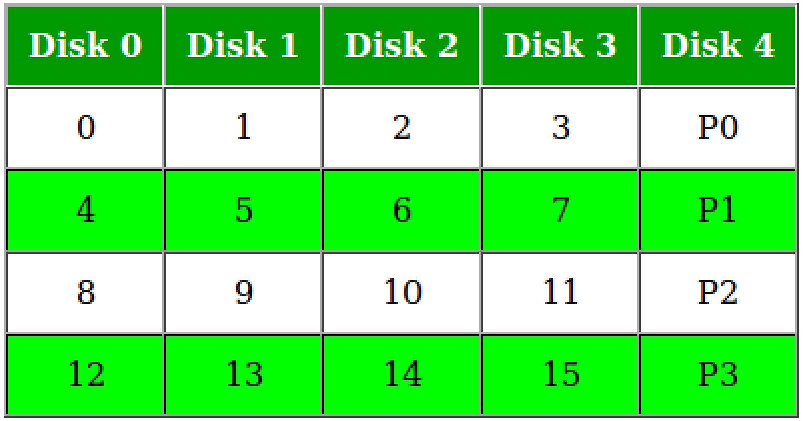
- *In **RAID 3**, configuration data are divided into individual bytes (**byte-level striping**) and then saved on data disks, it also uses a dedicated disk "(**parity disk)**" to store parity.*
- *In case of failure, it allows to recover data by an **appropriate calculation** of the **remaining bytes** and **parity bytes** that correspond with them. A **single parity bit** is all that is needed to recover/detect the lost data from an array of disks.*
- *Less expensive in number of **extra disk required**.*

RAID-3 (Bit - interleaved Parity)

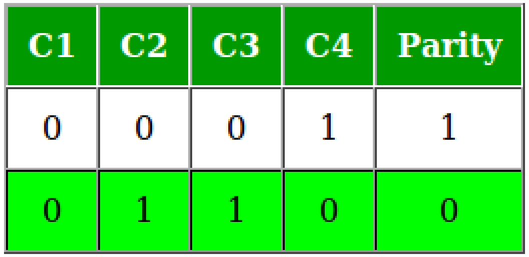


Parity disk

RAID-3 (Bit - interleaved Parity)



Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
4	5	6	7	P1
8	9	10	11	P2
12	13	14	15	P3



C1	C2	C3	C4	Parity
0	0	0	1	1
0	1	1	0	0

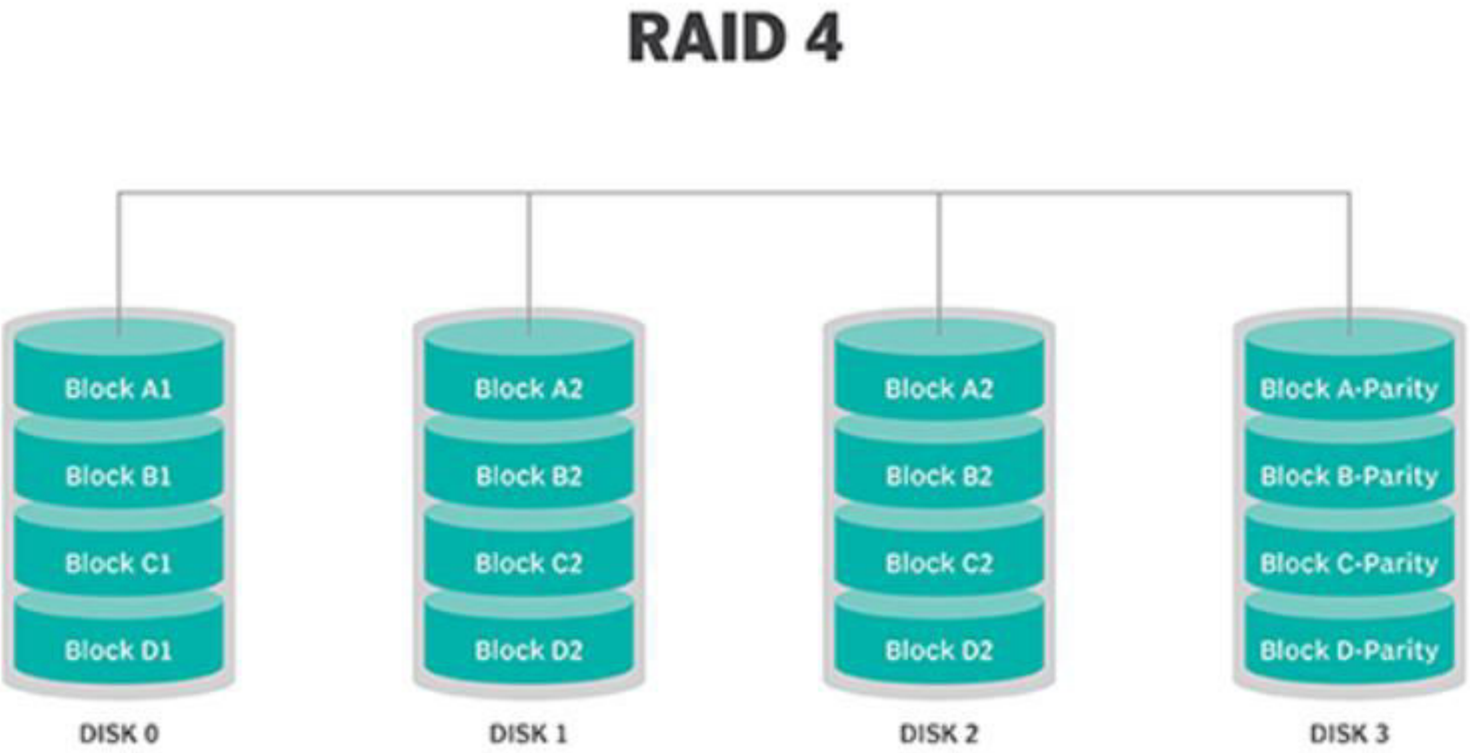
Parity is calculated using a simple XOR function. If the data bits are ***0,0,0,1*** *the parity bit is XOR(0,0,0,1) = 1. If the data bits are* ***0,1,1,0*** *the parity bit is* ***XOR(0,1,1,0)*** *= 0. A simple approach is that even number of ones results in parity 0, and an odd number of ones results in parity 1.*

Assume that in the above figure, C3 is lost due to some disk failure. Then, we can recompute the data bit stored in C3 by looking at the values of all the other columns and the parity bit. This level allows us to recover lost data.

RAID-4 (Block - interleaved Parity)

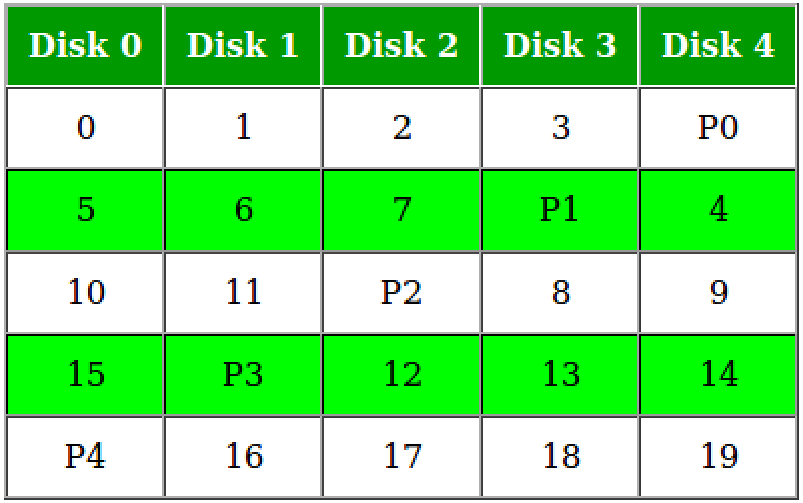
- ***RAID 4*** *is very similar to* ***RAID 3****. The main difference is the way of sharing data. They are divided* ***into blocks*** *(16, 32, 64 or 128 kB) and written on disk s – similar to RAID.*
- ***RAID 4*** *uses block level stripping rather than byte level with a parity disk. If a data disk fails, the parity data is used to create a replacement disk.*
- *Instead of duplicating data, the RAID 4 adopts a* ***parity-based*** *approach.*
- *This level allows recovery of at most* ***1 disk failure*** *due to the way* ***parity*** *works. In this level, if more than one disk fails, then there is no way to recover the data.*
- *Level 3 and level 4 both are required at least* ***three disks*** *to implement RAID and max is controller dependent.*

RAID-4 (Block - interleaved Parity)



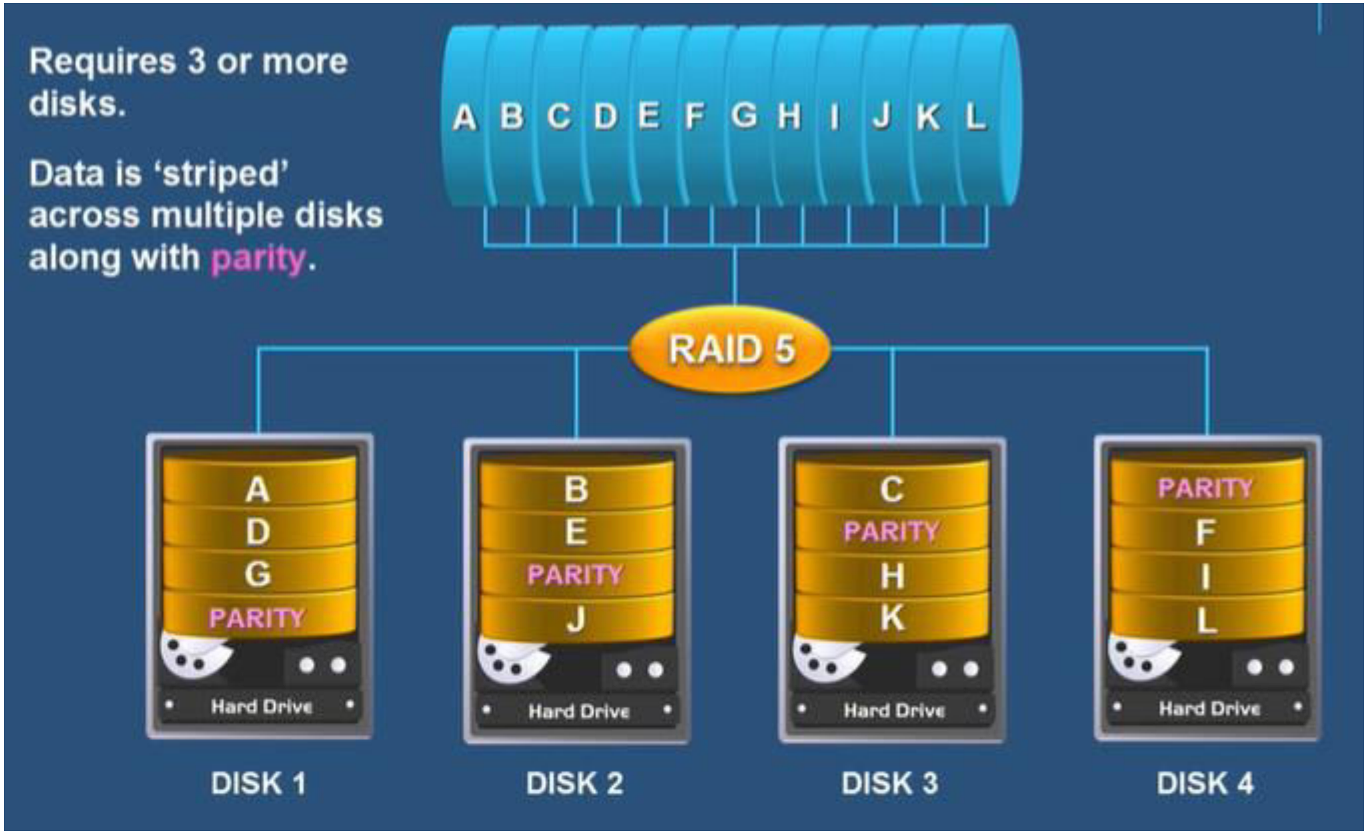
RAID-5 (Block - Interleaved Distributed Parity)

- *This level is similar to **level 4**, except the **parity blocks are distributed** over all disks, thereby more evenly balancing the load on the system. **Parity rotates among the drives.***
- *Note that the same disk cannot hold both data and parity for the same block, as both would be lost in the event of a disk crash.*
- *Most popular and cost effective solution.*



Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

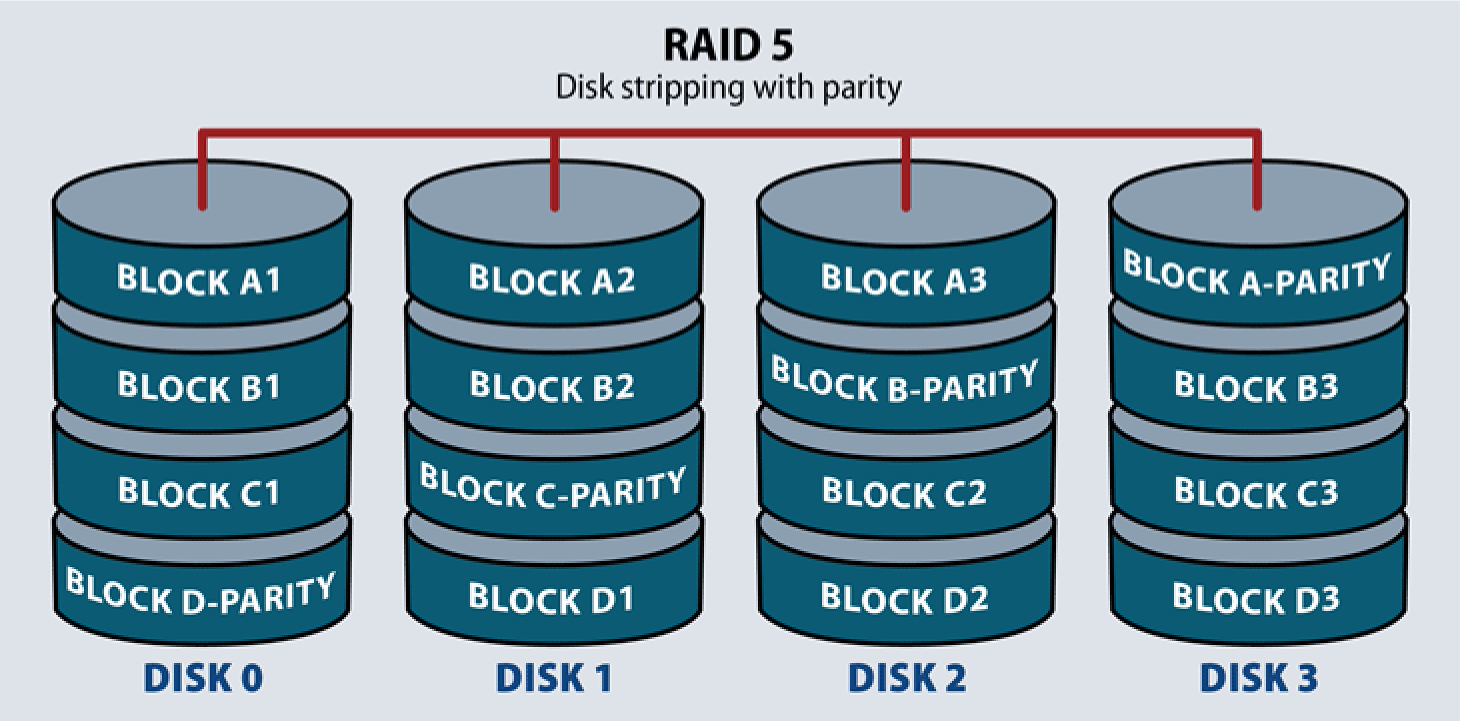
RAID-5 (Block - interleaved distributed parity)


Requires 3 or more disks.

Data is 'striped' across multiple disks along with parity.

RAID-5 (Block - interleaved distributed parity)

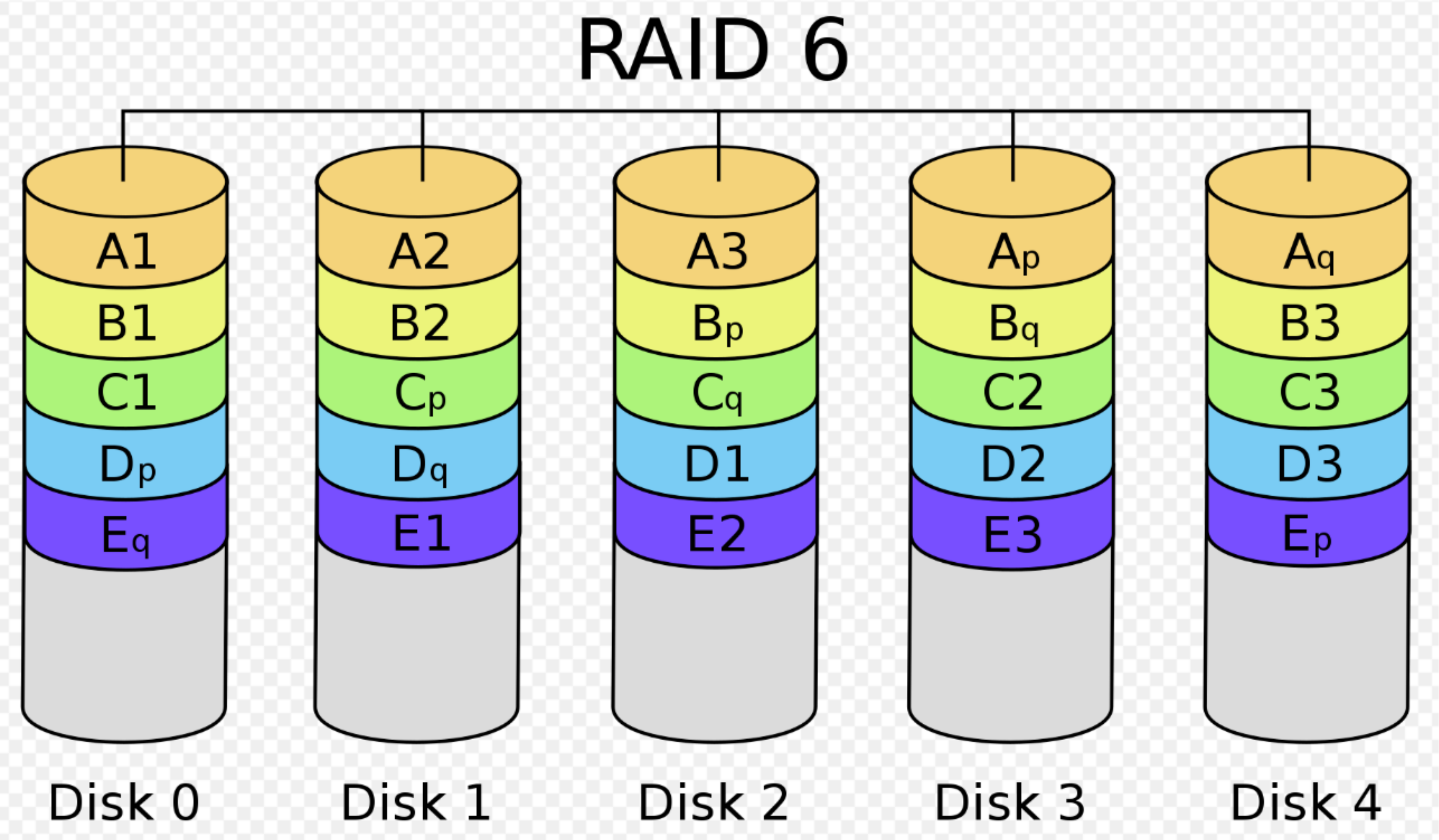
- *RAID 5 can withstand the loss of one disk in the array.*
- *This level increases write performance since all drives in the array simultaneously serve write requests.*



RAID-6 (P + Q Redundency)

- *This level is an **enhanced version** of RAID 5 adding extra benefit of **dual parity**.*
- *This level uses **block level stripping** with **DUAL distributed parity**. So now you can get extra redundancy.*
- *Imagine you are using RAID 5 and 1 of your disk fails so you need to hurry to replace the failed disk because if simultaneously another disk fails then you won't be able to recover any of the data so for those situations RAID 6 plays its part where you can survive 2 concurrent disk failures before you run out of options.*
- *In RAID 6, you can survive **2 concurrent disk failures**. However, this extra protection comes at a cost.*
- ***Horizontal** as well as **diagonal** parity generated.*
- *Min 4 disk and maximum controller dependent.*

RAID-6 (P + Q Redundency)



Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
A1	A2	A3	A_p	A_q
B1	B2	B_p	B_q	B3
C1	C_p	C_q	C2	C3
D_p	D_q	D1	D2	D3
E_q	E1	E2	E3	E_p

Disk Transfer Time

Taccess = Seek Time + Rotational Latency + Data Tranasfer time

*Seek time = no of seek * seek time (time for 1 track to next track(provided by manufacturer))*

*Transfer rate = no of heads * capacity of track * no of rotations per second*

Disk Transfer Time

Consider a disk having 16 platters, 512 tracks/surface , 2k sector/track and sector size is 4 kb, what is I/O time for 1 sector (Avg seek time is 30ms) disk rotation is 3600 rpm calculate effective data transfer time.