Very very - ---- ∞ Important

6.Disk Scheduling or I/O Scheduling, Different Disk Scheduling Algorithms

Disk scheduling is a technique which decide the order of service of the requests for disk. Disk scheduling is done by operating systems to schedule I/O requests arriving for disk. Disk scheduling is also known as I/O scheduling.

Disk scheduling is important because:

- 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 waiting queue and need to be scheduled.
- Two or more request may be far from each other so can result in greater disk arm movement.
- Hard drives are one of the slowest parts of computer system and thus need to be accessed in an efficient manner.

There are many Disk Scheduling Algorithms but before discussing them let's have a quick look at some of the important terms:

Seek Time: Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.

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 or it is the additional time taken to reach at desired sector after reaching the track which contains that sector.So the disk scheduling algorithm that gives minimum rotational latency is better.

Transfer Time: Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.

Disk Access Time or Service Time:

Disk Access Time or service time = Seek Time + Rotational Latency +Transfer Time

Disk bandwidth: 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 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 the all requests.

Variance Response Time is measure of how individual request are serviced with respect to average response time. So the disk scheduling algorithm that gives minimum variance response time is better.

Several algorithms exist to schedule the servicing of disk I/O requests.

We illustrate them with a request queue (0-199). 98, 183, 37, 122, 14, 124, 65, 67 Head pointer 53

(1) FCFS (First Come First Serve)

FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue. (Serviced)

Advantages:

- Every request gets a fair chance.
- No indefinite postponement.

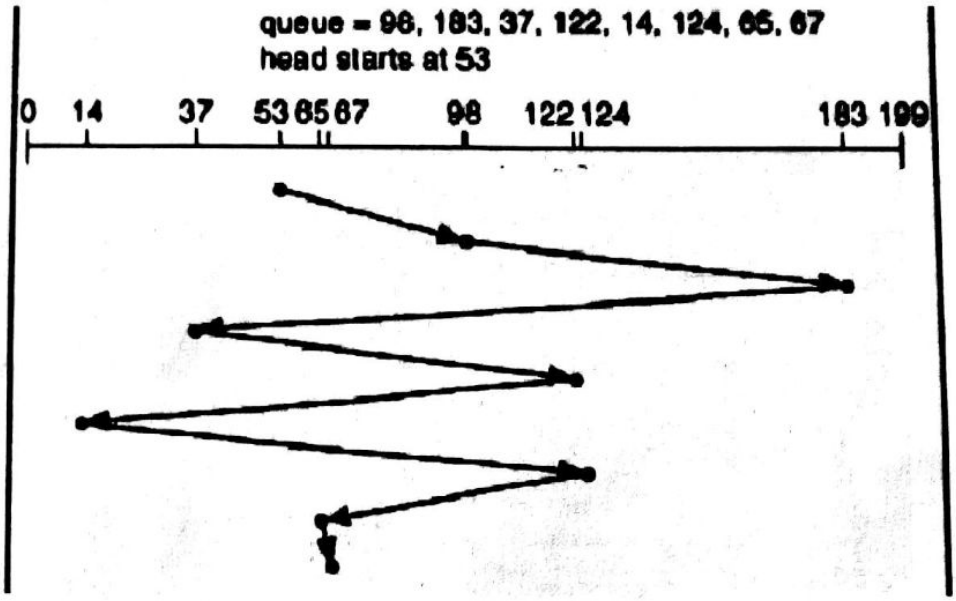
Disadvantages:

- Does not try to optimize seek time.
- May not provide the best possible service.

Example : Consider requests for blocks on the following cylinders

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

– Assume the disk head is initially on cylinder 53
  - The next slide shows the head traversal



Total disk head movement is 640 cylinders

(2) SSTF (Shortest Seek Time First)

In SSTF (Shortest Seek Time First), requests having shortest seek time are executed first. So, the seek time of every request is calculated in advance in 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.

SSTF is certainly an improvement over FCFS as it decreases the average response time and increases the throughput of system.

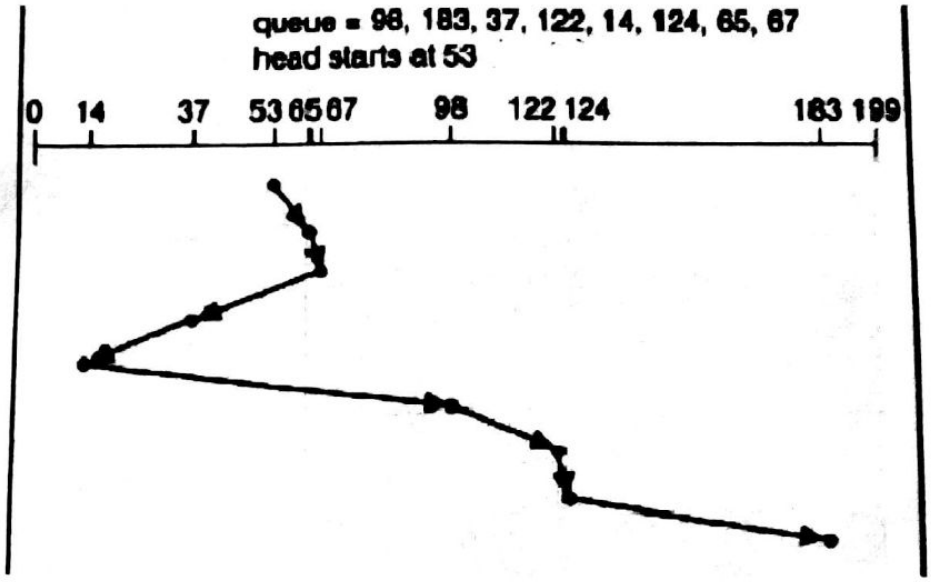
Advantages:

- Average Response Time decreases.
- Throughput increases.

Disadvantages:

- Overhead to calculate seek time in advance
- Can cause Starvation for a request if it has higher seek time as compared to incoming requests
- High variance of response time as SSTF favours only some requests

Example : Next slide depicts head movement for previous example



Total head movement or seek time is 236 cylinder.

(3) Elevator algorithm (SCAN Algorithm)

In SCAN algorithm the disk arm moves into a particular direction (either increasing order of track no or decreasing order of track no) from current head position and services the requests coming in its path and after reaching the end of disk (Last track in that direction), it reverses its direction and again services the remaining request arriving in its path and move till it reaches the last request(Submit in queue) to be service in that direction. So, this algorithm works like an elevator and hence also known as elevator algorithm. As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

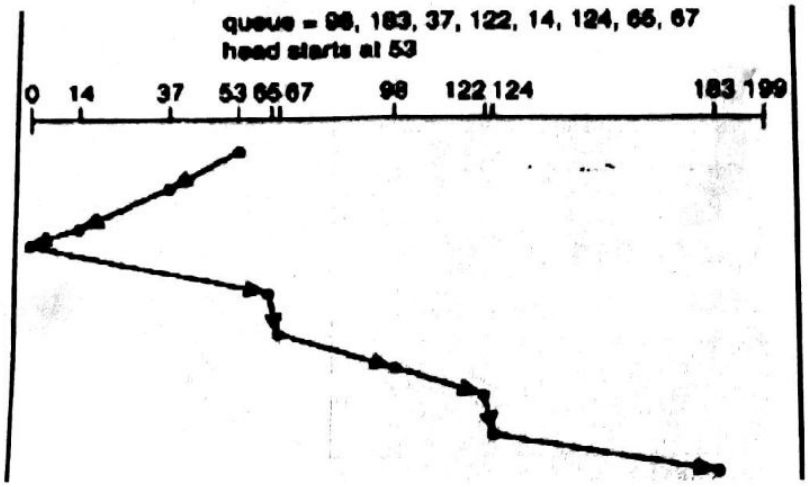
Advantages:

- High throughput
- Low variance of response time
- Average response time

Disadvantages:

- Long waiting time for requests for locations just visited by disk arm

Next slide depicts head movement for previous example



Total head movement is 208 cylinders (when the disk arm or read write head is moving in decreasing order of track number)

(4) C SCAN

In CSCAN (Circular Scan) disk arm moves into a particular direction (either increasing order of track no or decreasing order of track no) from current head position and services the requests coming in its path and after reaching the end of disk (Last track in that direction for eg track no 199 in figure), it reverses its direction and direct come back to other end of the disk (e.g track no 0 in the figure) and after that again reverse its direction and services the remaining request and head moves upto last remaining request.

Example – Consider a disk with 200 tracks (0-199) and that the disk request queue has random requests in it. The following track requests for I/O to blocks on cylinders:

60, 143, 15, 185, 85, 120, 33, 28, 146

Consider that the read/write head is initially at cylinder 70. Compute the total head movements.

Solution is shown in following figure

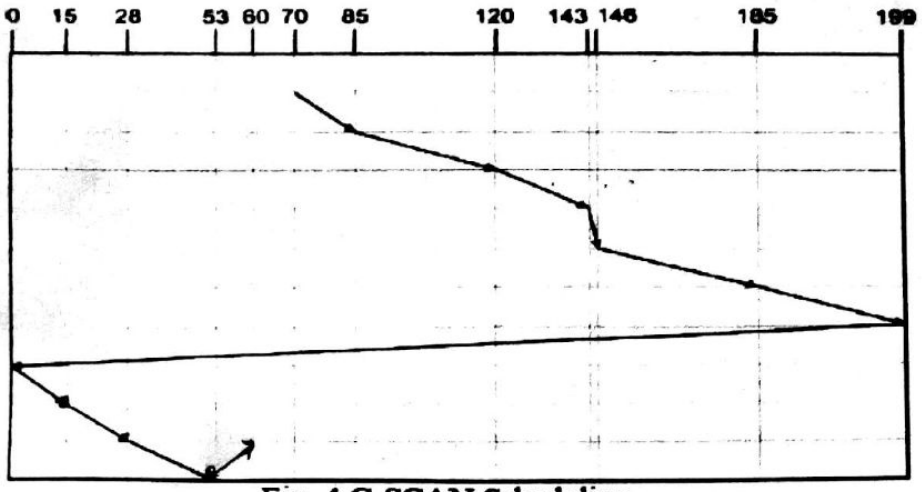


Fig. 4 C-SCAN Scheduling

Total Head Movements = (199-70) + (199-0) + (60-0) = 388 cylinders

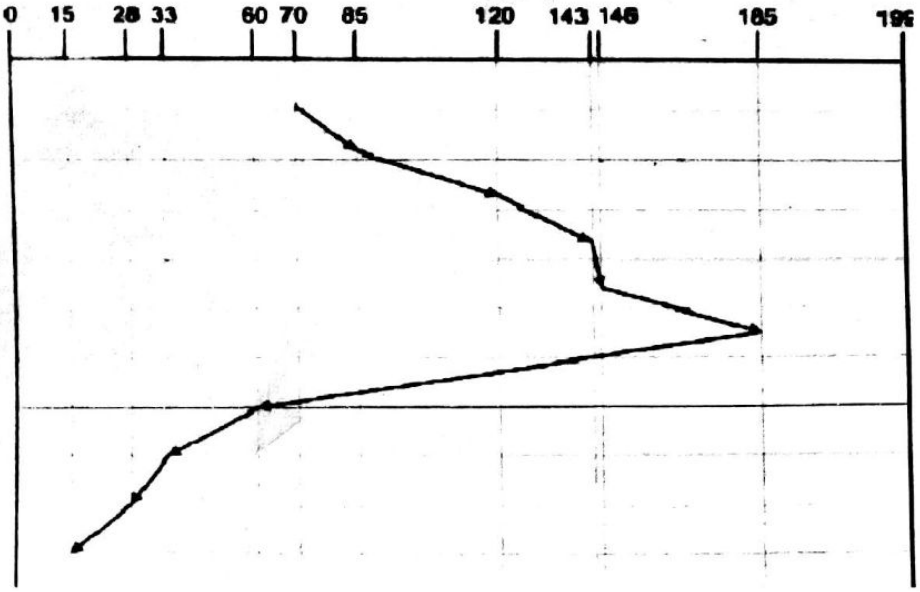
Advantages:

Provides more uniform wait time compared to SCAN.

(5) LOOK – In LOOK algorithm the disk arm moves into a particular direction (either increasing order of track no or decreasing order of track no) from current head position and services the requests coming in its path and moves till the last request to be service(e.g request for track no 185 in figure) in that direction and then it reverses its direction and again services the remaining request arriving in its path and move till it reaches the last request(e.g request for track no 15 in the following example) in that direction to be service in that direction.

Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Example – for above example



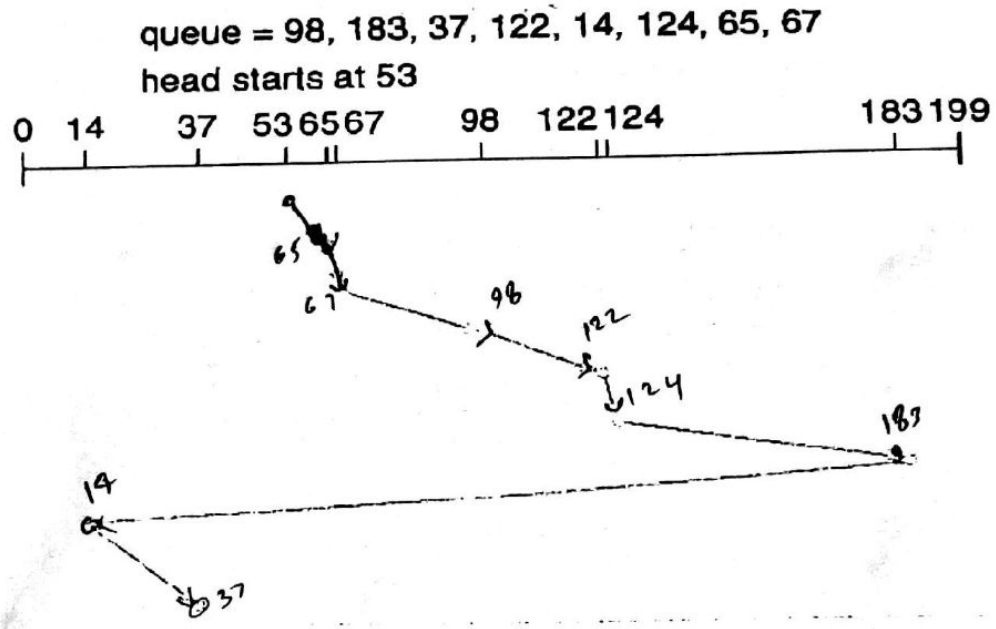
Total Head Movements = (185-70) + (185-15)= 285 cylinders.

(6) C-LOOK:

As LOOK is similar to SCAN algorithm, in similar way, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm inspite of going to the end or last track , it simply goes only to the last request to be serviced in moving direction (for eg 183 in following figure) and then from there it reverse it direction and direct goes to the other last request in that direction(e.g 14 in figure) without service the request and then reverse its direction and service the remaining requests.

Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Example – As shown in following figure



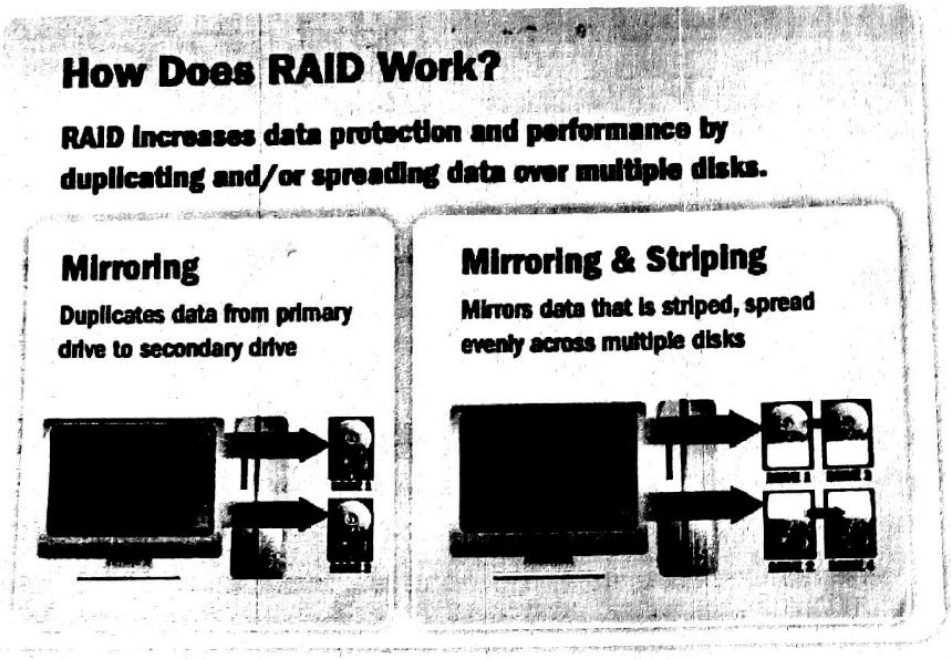
7.Selecting a Disk Scheduling Algorithm

- SSTF is common and has a natural appeal
- Elevator and C-SCAN perform better for systems that place a heavy load on the disk
- Performance depends on the number and types of requests .
- Requests for disk service can be influenced by the file allocation method
- 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 SSTF or Elevator is a reasonable choice for the default algorithm.

8.RAID

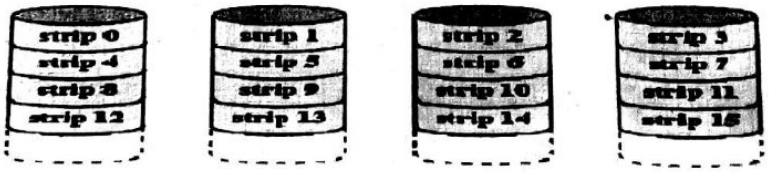
RAID stands for Redundant Array of Inexpensive Disks. RAID is a method of logically treating several hard drives as one unit. It can offer fault tolerance and higher throughput levels than a single hard drive or group of independent hard drives.

Why Do We Need It? RAID provides real-time data recovery when a hard drive fails, increasing system uptime and availability while protecting against loss of data. Multiple drives working together also increase system performance.



Types or Levels of RAID

RAID 0 - Stripped

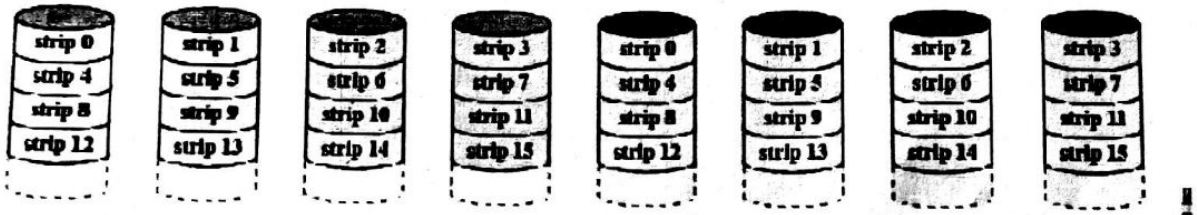


(a) RAID 0 (non-redundant)

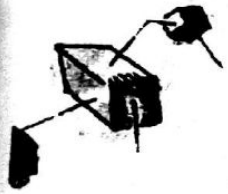
- Not a true RAID – no redundancy
- Disk failure is catastrophic
- Very fast due to parallel read/write

RAID 1 - Mirrored

- Redundancy through duplication instead of parity.
- Read requests can made in parallel.
- Simple recovery from disk failure

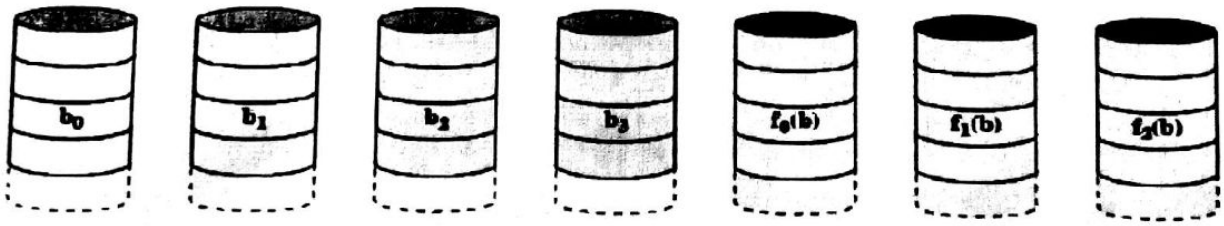


(b) RAID 1 (mirrored)



RAID 2
(Using Hamming code)

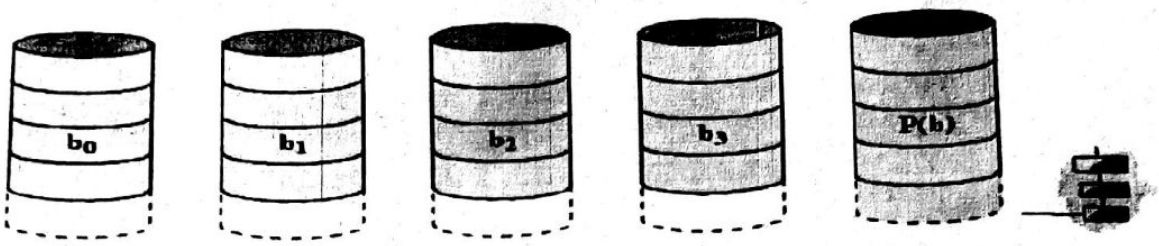
- Synchronised disk rotation
- Data stripping is used (extremely small)
- Hamming code used to correct single bit errors and detect double-bit errors



(c) RAID 2 (redundancy through Hamming code)

RAID 3
bit-interleaved parity

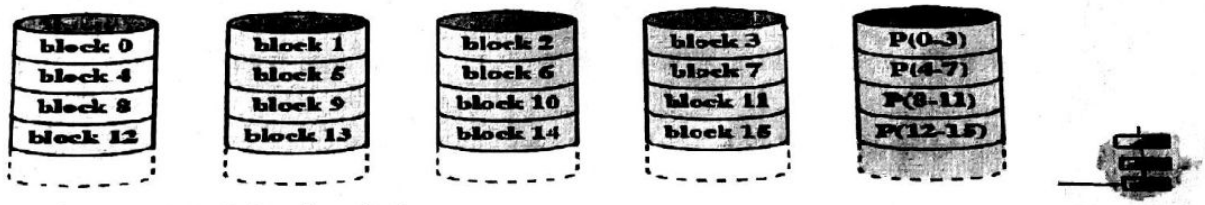
- Similar to RAID-2 but uses all parity bits stored on a single drive



(d) RAID 3 (bit-interleaved parity)

RAID 4
Block-level parity

- A bit-by-bit parity strip is calculated across corresponding strips on each data disk
- The parity bits are stored in the corresponding strip on the parity disk.

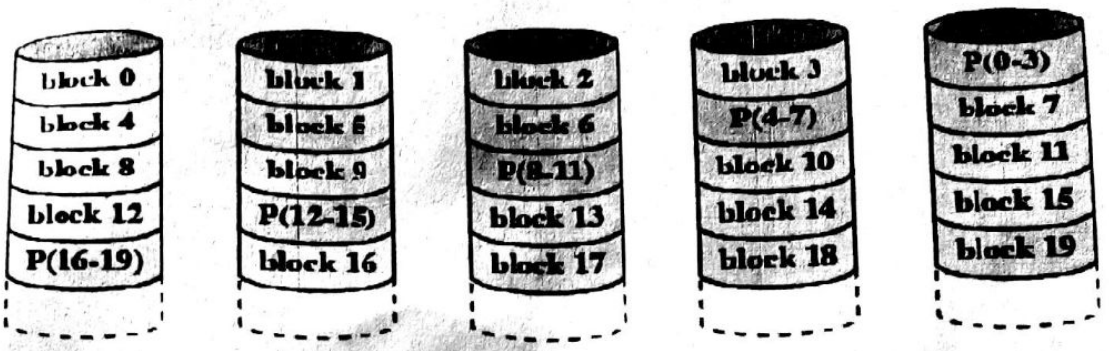


(e) RAID 4 (block-level parity)

RAID 5

Block-level Distributed parity

- Similar to RAID-4 but distributing the parity bits across all drives

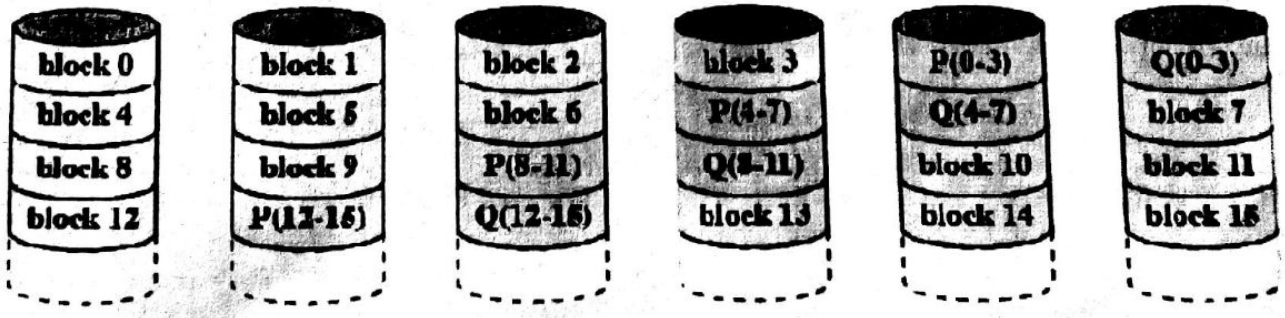


(f) RAID 5 (block-level distributed parity)

RAID 6

Dual Redundancy

- Two different parity calculations are carried out
  - stored in separate blocks on different disks.
- Can recover from two disks failing



(g) RAID 6 (dual redundancy)