1 Mass-Storage Structure

- Management and ordering of disk access requests is important:
  - Huge speed gap between memory and disk.
  - Disk throughput is extremely sensitive to.
    - Request order $\Longrightarrow$ **Disk Scheduling**.
    - Placement of data on the disk $\Longrightarrow$ **File System Design**.
  - Disk scheduler must be aware of *disk geometry*.
- Disk management issues
  - Formatting
    - *Physical*: divide the blank slate into sectors identified by headers containing such information as sector number.
    - *Logical*: marking bad blocks; partitioning and writing a blank directory on disk; installing file allocation tables, and other relevant information (file system initialization).
  - Reliability
    - RAIDs (Redundant Array of Inexpensive Disks): various levels, level 0 is disk striping).

1.1 Overview of Mass-Storage Structure

1.1.1 Magnetic Disks

- **Magnetic disks** provide the bulk of secondary storage for modern computer systems. Conceptually, disks are relatively simple (see Fig. 1).
  - Each disk **platter** has a flat circular shape, like a CD.
  - Common platter diameters range from 1.8 to 5.25 inches.
  - The two surfaces of a platter are covered with a magnetic material. We store information by recording it magnetically on the platters.
  - A read-write head "flies" just above each surface of every platter.
  - The heads are attached to a disk arm that moves all the heads as a unit.
  - The surface of a platter is logically divided into circular **tracks**, which are subdivided into **sectors**.

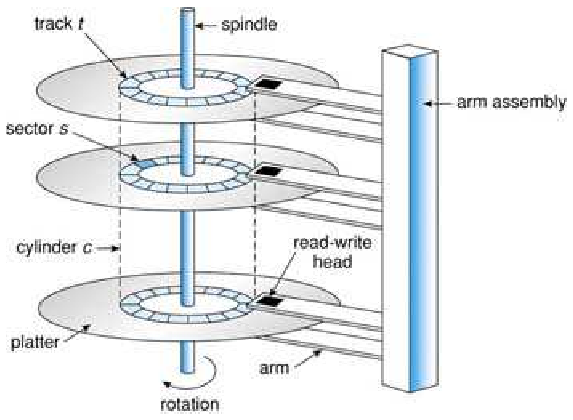


Figure 1: Moving-head disk mechanism.

- The set of tracks that are at one arm position makes up a **cylinder**.
- There may be thousands of concentric cylinders in a disk drive, and each track may contain hundreds of sectors.

- When the disk is in use, a drive motor spins it at high speed. Most drives rotate 60 to 200 times per second.
- Execution of a disk operation involves (see Fig. 2)

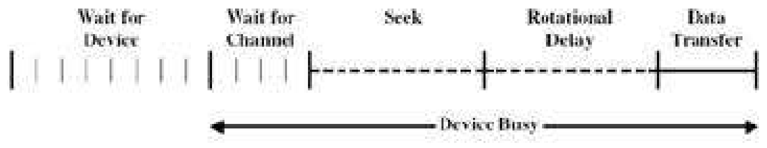


Figure 2: Disk Performance.

1. **Wait time**: the process waits to be granted device access;
   - *Wait for device*: time the request spend in wait queue.
   - *Wait for channel*: time until a shared I/O channel is available.
2. **Positioning time**: time hardware need to position the head. Sometimes called the **random-access time**,
   - consists of the time to move the disk arm to the desired cylinder, called the **seek time** T_s (in milliseconds),

  - the time for the desired sector to rotate to the disk head, called the **rotational latency** (in milliseconds). Rotational speed, r, of 5000 to 10000 rpm.

3. **Transfer rate** is the rate at which data flow between the drive and the computer (megabytes of data per second). **Transfer time**: to transfer b bytes, with N bytes per track;

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

- **Total average access time**;

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

- A timing comparison for $T_s = 2$ ms, $r = 10000$rpm, 512B sector size, 320 sectors per track, 1.3 MB file size.
  - At 10000 rpm, one revolution per 6ms⇒ average delay 3ms (=(60 second/10000) & $\frac{1}{r} = 6$).
  - Read a file with 2560 sectors (=(1.3MB/512))
  - File stored compactly (8 adjacent tracks (=(2560/320))). Read first track;

Average seek	2ms
Rot. Delay	3ms
Read 320 sectors	6ms ($\frac{1}{r} = 6$, b=512*320 & N=512*320)
Total	11ms
All sectors	11+7*9=74ms

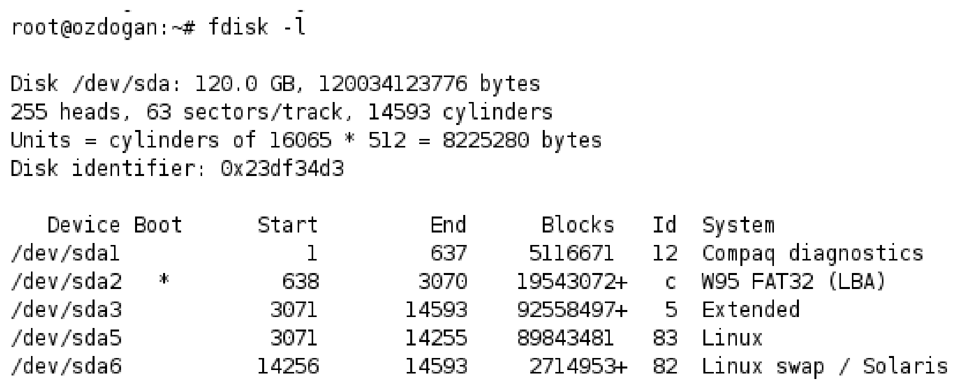
  - Sectors distributed randomly over the disk: Read any sector

Average seek	2ms
Rot. Delay	3ms
Read 1 sectors	0.01875ms (=(6/320); $\frac{1}{r} = 6$, b=512 , N=512*320)
Total	5.01875ms
All	2560*5.01875=12848ms

- Disk Performance is entirely dominated by *Seek* and *Rotational Delays.*
  - It will only get worse as capacity increases much faster than increase in seek time and rotation speed.

- It has been easier to spin the disk faster than improve seek time.

- **Floppy disks** are inexpensive removable magnetic disks that have a soft plastic case containing a flexible platter.
  - The head of a floppy-disk drive generally sits directly on the disk surface, so the drive is designed to rotate more slowly than a hard-disk drive to reduce the wear on the disk surface.
- 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
  - enhanced integrated drive electronics (EIDE),
  - advanced technology attachment (ATA),
  - serial ATA (SATA),
  - universal serial bus (USB),
  - fiber channel (FC),
  - SCSI buses
- 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.
    - To perform a disk I/O operation, the computer places a command into the host controller (memory-mapped I/O ports).
    - The host controller then sends the command via messages to the disk controller.
    - The disk controller operates the disk-drive hardware to carry out the command.
  - A **disk controller** is built into each disk drive.
    - Disk controllers usually have a built-in cache.
    - Data transfer at the disk drive happens
      - between the cache and the disk surface,
      - between the cache and the host controller.



```
root@ozdogan:~# fdisk -l

Disk /dev/sda: 120.0 GB, 120034123776 bytes
255 heads, 63 sectors/track, 14593 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x23df34d3

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1               1         637     5116671   12  Compaq diagnostics
/dev/sda2   *         638        3070    19543072+   c  W95 FAT32 (LBA)
/dev/sda3            3071       14593    92558497+   5  Extended
/dev/sda5            3071       14255    89843481   83  Linux
/dev/sda6           14256       14593     2714953+  82  Linux swap / Solaris
```

Figure 3: System's partition table.

1.2 Disk Structure

- Modern disk drives are addressed as large one-dimensional arrays of *logical blocks*, where the logical block is the smallest unit of transfer.
- The size of a logical block is usually 512 bytes, although some disks can be low-level formatted to have a different logical block size, such as 1,024 bytes.
- The one-dimensional array of logical blocks is mapped onto the sectors of the disk sequentially.
  - Sector 0 is the first sector of the first track on the outermost cylinder.
  - The mapping proceeds in order through that track, then through the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.
- By using this mapping, we can convert a logical block number into an old-style disk address that consists of a cylinder number, a track number within that cylinder, and a sector number within that track.
- In practice, it is difficult to perform this translation, for two reasons.
  - First, most disks have some defective sectors, but the mapping hides this by substituting spare sectors from elsewhere on the disk.
  - Second, the number of sectors per track is not a constant on some drives.

- Let's look more closely at the second reason.
  - On media that use **constant linear velocity** (CLV), the density of bits per track is uniform.
  - The farther a track is from the center of the disk, the greater its length, so the more sectors it can hold.
    - As we move from outer zones to inner zones, the number of sectors per track decreases (see Fig. 4).
    - 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 (CD-ROM, DVD-ROM drives).

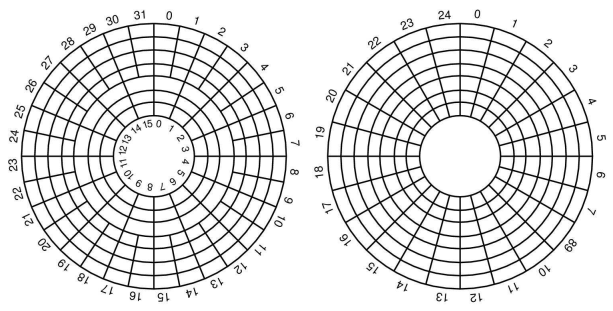
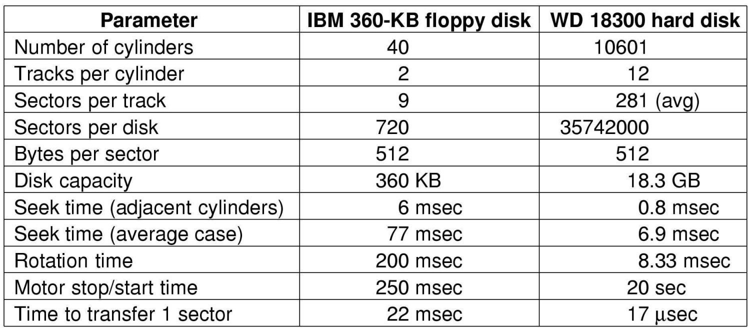


Figure 4: Physical geometry of a disk with two zones and a possible virtual geometry for this disk.

  - Alternatively, the disk rotation speed can stay constant, and 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)**.

- Evolution of Disk Hardware (see Fig. 5)
  - Average seek time is approx 12 times better.
  - Rotation time is 24 times faster.
  - Transfer time is 1300 times faster.
  - Most of this gain is due to increase in density.



Parameter	IBM 360-KB floppy disk	WD 18300 hard disk
Number of cylinders	40	10601
Tracks per cylinder	2	12
Sectors per track	9	281 (avg)
Sectors per disk	720	35742000
Bytes per sector	512	512
Disk capacity	360 KB	18.3 GB
Seek time (adjacent cylinders)	6 msec	0.8 msec
Seek time (average case)	77 msec	6.9 msec
Rotation time	200 msec	8.33 msec
Motor stop/start time	250 msec	20 sec
Time to transfer 1 sector	22 msec	17 µsec

Figure 5: Disk parameters for the original IBM PC floppy disk and a Western Digital WD 18300 hard disk.

1.3 Disk Attachment

Computers access disk storage in two ways.

- One way is via I/O ports (or host-attached storage); this is common on small systems.
- The other way is via a remote host in a distributed file system; this is referred to as network-attached storage.

1.3.1 Host-Attached Storage

- Host-attached storage is storage accessed through local I/O ports.
- These ports use several technologies.
  - The typical desktop PC uses an I/O bus architecture called IDE or ATA. This architecture supports a maximum of two drives per I/O bus.
  - A newer, similar protocol that has simplified cabling is SATA.
  - High-end workstations and servers generally use more sophisticated I/O architectures, such as SCSI and fiber channel (FC).
- **SCSI** is a bus architecture;
  - The SCSI protocol supports a maximum of 16 devices on the bus. Generally, the devices include one controller card in the host (the SCSI initiator) and up to 15 storage devices (the SCSI targets).

  - A SCSI disk is a common SCSI target, but the protocol provides the ability to address up to 8 logical units in each SCSI target.
  - A typical use of logical unit addressing is to direct commands to components of a RAID array.

- **FC** is a high-speed serial architecture that can operate over optical fiber or over a four-conductor copper cable.

- A wide variety of storage devices are suitable for use as host-attached storage. Among these are hard disk drives, RAID arrays, and CD, DVD, and tape drives.

- The I/O commands that initiate data transfers to a host-attached storage device are reads and writes of logical data blocks directed to specifically identified storage units (such as bus ID, SCSI ID, and target logical unit).

1.3.2 Network-Attached Storage

- A network-attached storage (NAS) device is a special-purpose storage system that is accessed remotely over a data network (see Fig. 6).

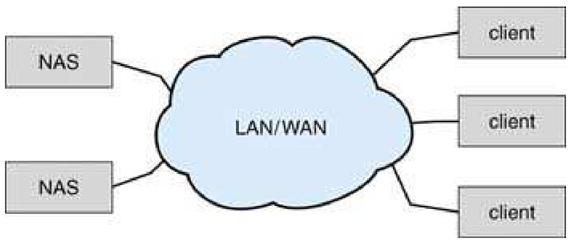


Figure 6: Network-attached storage.

- Clients access network-attached storage via a remote-procedure-call interface such as NFS for UNIX systems or CIFS for Windows machines.

- The remote procedure calls (RPCs) are carried via TCP or UDP over an IP network -usually the same local-area network (LAN) that carries all data traffic to the clients.

- Network-attached storage provides a convenient way for all the computers on a LAN to share a pool of storage with the same ease of naming and access enjoyed with local host-attached storage. However, it tends to be <u>less efficient</u> and have <u>lower performance</u> than some direct-attached storage options.

- ISCSI is the latest network-attached storage protocol. In essence, it uses the IP network protocol to carry the SCSI protocol.
- Thus, networks rather than SCSI cables can be used as the interconnects between hosts and their storage. As a result, hosts can treat their storage as if it were directly attached, but the storage can be distant from the host.

1.3.3 Storage-Area Network

- One drawback of network-attached storage systems is that the storage I/O operations <u>consume bandwidth</u> on the data network, thereby increasing the <u>latency</u> of network communication.
- A storage-area network (SAN) is a private network (using storage protocols rather than networking protocols) connecting servers and storage units, as shown in Fig. 7.

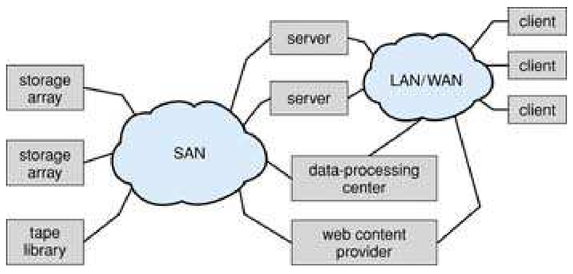


Figure 7: Storage-area network.

- The power of a SAN lies in its flexibility. Multiple hosts and multiple storage arrays can attach to the same SAN, and storage can be dynamically allocated to hosts.
- A SAN switch allows or prohibits access between the hosts and the storage. As one example, if a host is running low on disk space, the SAN can be configured to allocate more storage to that host.
- SANs typically have more ports, and less expensive ports, than storage arrays.

- FC is the most common SAN interconnect. An emerging alternative is a special-purpose bus architecture named InfiniBand, which provides hardware and software support for high-speed interconnection networks for servers and storage units.

1.4 Disk Scheduling

- One of the responsibilities of the OS is to use the hardware efficiently. For the disk drives, meeting this responsibility entails having fast access time and large disk bandwidth.
- The access time has two major components.
  - The **seek time** is the time for the disk arm to move the heads to the cylinder containing the desired sector.
  - 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.
- We can improve both the access time and the bandwidth by scheduling the servicing of disk I/O requests in a good order.
- For a multiprogramming system with many processes, the disk queue may often have several pending requests.
  - Thus, when one request is completed, the OS chooses which pending request to service next.
  - How does the OS make this choice? Disk-scheduling algorithms.

1.4.1 FCFS Scheduling

- The simplest form of disk scheduling is the first-come, first-served (FCFS) algorithm. This algorithm is intrinsically fair, but it generally does not provide the fastest service.
- Consider, for example, a disk queue with requests for I/O to blocks on cylinders in that order;

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

- If the disk head is initially at cylinder 53,
- It will first move from 53 to 98,
- then to 183, 37, 122, 14, 124, 65, and finally to 67,
- for a total head movement of 640 cylinders.
- This schedule is diagrammed in Fig. 8.

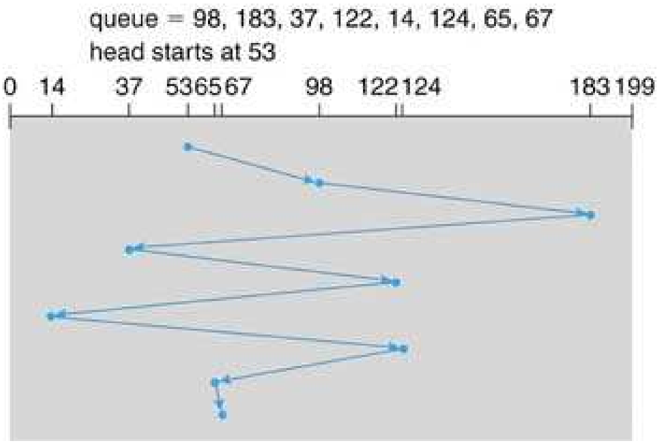


Figure 8: FCFS disk scheduling.

- The wild swing from 122 to 14 and then back to 124 illustrates the problem with this schedule.
- If the requests for cylinders 37 and 14 could be serviced together, before or after the requests at 122 and 124, the total head movement could be decreased substantially, and performance could be thereby improved.

1.4.2 SSTF Scheduling

- It seems reasonable to service all the requests close to the current head position before moving the head far away to service other requests. This assumption is the basis for the shortest-seek-time-first (SSTF) algorithm.
- The SSTF algorithm selects the request with the minimum seek time from the current head position.
- Since seek time increases with the number of cylinders traversed by the head, SSTF chooses the pending request closest to the current head position.

- For our example request queue (see Fig. 9);

```
53, 65, 67, 37, 14, 98, 122, 124, 183
```

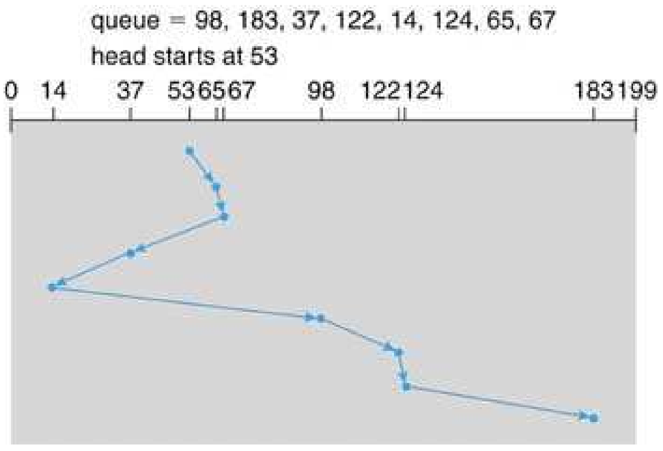


Figure 9: SSTF disk scheduling.

- This scheduling method results in a total head movement of only 236 cylinders-little more than one-third of the distance needed for FCFS scheduling of this request queue. This algorithm gives a substantial improvement in performance.
- SSTF scheduling is essentially a form of shortest-job-first (SJF) scheduling; and like SJF scheduling, it may cause starvation of some requests (steady supply of shorter seek time requests).
- Although the SSTF algorithm is a substantial improvement over the FCFS algorithm, it is not optimal. Consider;

```
53, 37, 14, 65, 67, 98, 122, 124, 183
```

- This strategy reduces the total head movement to 208 cylinders.

1.4.3 SCAN Scheduling

- In the SCAN algorithm, the disk arm starts at one end of the disk and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk.

- At the other end, the direction of head movement is reversed, and servicing continues. The head continuously scans back and forth across the disk.
- The SCAN algorithm is sometimes called the elevator algorithm, since the disk arms behaves just like an elevator in a building, first servicing all the requests going up and then reversing to service requests the other way.
- For our example request queue, we need to know the direction of head movement in addition to the head's current position, 53 (see Fig. 9);

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

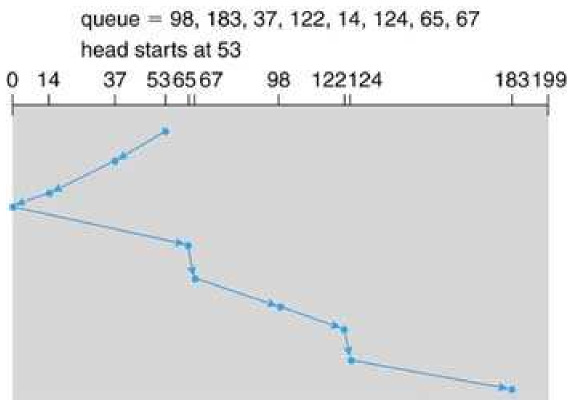


Figure 10: SCAN disk scheduling.

- If a request arrives in the queue just in front of the head, it will be serviced almost immediately
- If a request arriving just behind the head will have to wait until the arm moves to the end of the disk, reverses direction, and comes back.
- Assuming a uniform distribution of requests for cylinders, consider the density of requests when the head reaches one end and reverses direction.
  - At this point, relatively few requests are immediately in front of the head, since these cylinders have recently been serviced.
  - The heaviest density of requests is at the other end of the disk.

- These requests have also waited the longest, so why not go there first?
- That is the idea of the next algorithm.

1.4.4 C-SCAN Scheduling

- Circular SCAN (C-SCAN) scheduling is a variant of SCAN designed to provide a more uniform wait time.
- Like SCAN, CSCAN moves the head from one end of the disk to the other, servicing requests along the way.
- When the head reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip (see Fig. 11).

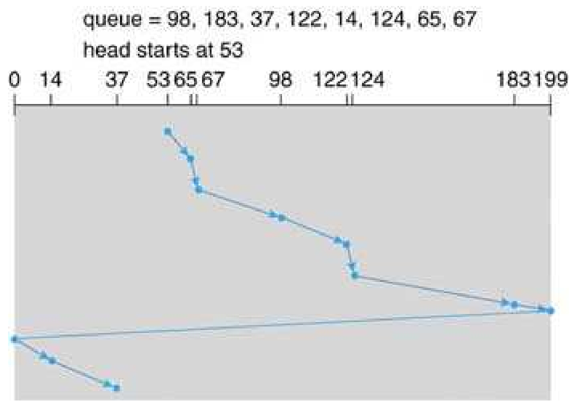


Figure 11: C-SCAN disk scheduling.

1.4.5 LOOK Scheduling

- As we described them, both SCAN and C-SCAN move the disk arm across the full width of the disk.
- In practice, neither algorithm is often implemented this way. More commonly, the arm goes only as far as the final request in each direction.
- Then, it reverses direction immediately, without going all the way to the end of the disk.

- Versions of SCAN and C-SCAN that follow this pattern are called LOOK and C-LOOK scheduling, because they look for a request before continuing to move in a given direction (see Fig. 12).

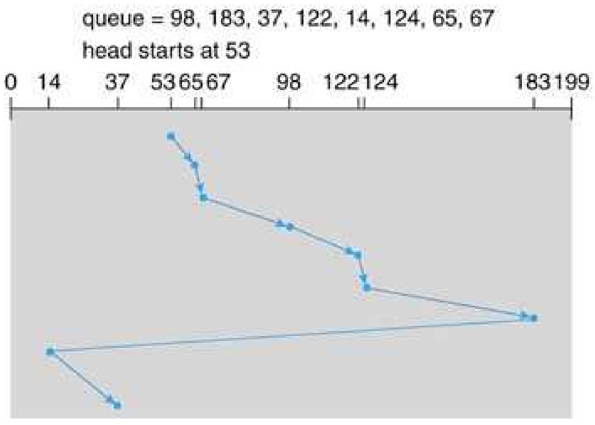


Figure 12: C-LOOK disk scheduling.

1.5 Disk Management

1.5.1 Disk Formatting

- A new magnetic disk is a blank slate: It is just a platter of a magnetic recording material.
- Before a disk can store data, it must be divided into sectors that the disk controller can read and write. This process is called **low-level formatting**, or **physical formatting**.
- Low-level formatting fills the disk with a special data structure for each sector.
  - The data structure for a sector typically consists of a header, a data area (usually 512 bytes in size), and a trailer.
  - The header and trailer contain information used by the disk controller, such as a sector number and an **error-correcting code** (ECC).
- When the controller **writes** a sector of data during normal I/O, the ECC is updated with a value calculated from all the bytes in the data area.

- When the sector is **read**, the ECC is recalculated and is compared with the stored value. 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 controller automatically does the ECC processing whenever a sector is read or written.
- To use a disk to hold files, the OS still needs to record its own data structures on the disk. It does so in two steps.
  - The first step is to partition the disk into one or more groups of cylinders.
    - The OS can treat each partition as though it were a separate disk.
    - For instance, one partition can hold a copy of the OS's executable code, while another holds user files.
  - After partitioning, the second step is **logical formatting** (or creation of a file system).
    - In this step, the OS 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.
- When reading sequential blocks, the seek time can result in missing block 0 in the next **track**. Disk can be formatted using a cylinder *skew* to avoid this (see Fig. 13).

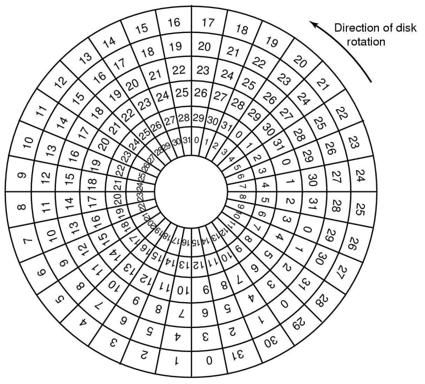


Figure 13: An illustration of cylinder skew.

- To increase efficiency, most 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.

1.6 RAID Structure

- A variety of disk-organization techniques, collectively called **redundant arrays of inexpensive disks** (**RAIDs**) are commonly used to address the performance and reliability issues.
- In the past, RAIDs composed of small, cheap disks were viewed as a cost-effective alternative to large, expensive disks.
- Today, RAIDs are used for their higher reliability and higher data-transfer rate, rather than for economic reasons. Hence, the *I* in RAID now stands for "independent" instead of "inexpensive".

1.6.1 Improvement of Reliability via Redundancy

- Suppose that the mean time to failure of a single disk is 100000 hours. Then the mean time to failure of some disk in an array of 100 disks will be $100000/100 = 1000$ hours, or 41.66 days, which is not long at all.
- If we store only one copy of the data, then each disk failure will result in loss of a significant amount of data-and such a high rate of data loss is unacceptable.
- The solution to the problem of reliability is to introduce redundancy; we store extra information that is not normally needed but that can be used in the event of failure of a disk to rebuild the lost information.
- Thus, even if a disk fails, data are not lost.
- The simplest (but most expensive) approach to introducing redundancy is to duplicate every disk. This technique is called **mirroring**.

1.6.2 Improvement in Performance via Parallelism

- With multiple disks, we can improve the transfer rate as well (or instead) by striping data across the disks.
- In its simplest form, data striping consists of splitting the bits of each byte across multiple disks; such striping is called bit-level striping.

  - For example, if we have an array of eight disks, we write bit i of each byte to disk i.

- The array of eight disks can be treated as a single disk with sectors that are eight times the normal size and, more important that have eight times the access rate.

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

  1. Increase the throughput of multiple small accesses (that is, page accesses) by load balancing.
  2. Reduce the response time of large accesses.

1.6.3 RAID Levels

- Mirroring provides high reliability, but it is expensive.

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

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

- These schemes have different cost-performance trade-offs and are classified according to levels called **RAID levels** (see Figs. 14); (in the figure, P indicates error-correcting bits, and C indicates a second copy of the data).

- In all cases depicted in the figure, four disks' worth of data are stored, and the extra disks are used to store redundant information for failure recovery.

- **RAID Level O**. RAID level 0 refers to disk arrays with striping at the level of blocks but without any redundancy (such as mirroring or parity bits).

- **RAID Level 1**. RAID level 1 refers to disk mirroring.

- **RAID Level 2**. RAID level 2 is also known as **memory-style error-correcting-code (ECC) organization**. Memory systems have long detected certain errors by using parity bits.

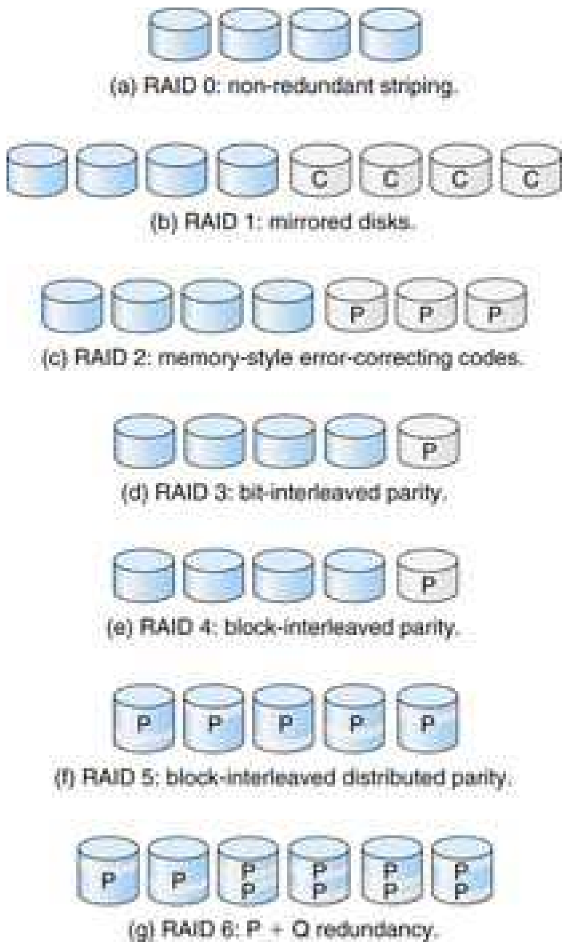

(a) RAID 0: non-redundant striping.
C
(b) RAID 1: mirrored disks.
P
(c) RAID 2: memory-style error-correcting codes.
(d) RAID 3: bit-interleaved parity.
(e) RAID 4: block-interleaved parity.
(f) RAID 5: block-interleaved distributed parity.
(g) RAID 6: P + Q redundancy.

Figure 14: RAID levels.

- **RAID Level 3**. RAID level 3, or **bit-interleaved parity organization**, improves on level 2 by taking into account the fact that, unlike memory systems, disk controllers can detect whether a sector has been read correctly, so a single parity bit can be used for error correction as well as for detection.
- **RAID Level 4**. RAID level 4, or block-interleaved parity organization, uses block-level striping, as in RAID 0, and in addition keeps a parity block on a separate disk for corresponding blocks from N other disks.
- **RAID Level 5**. RAID level 5, or block-interleaved distributed parity, differs from level 4 by spreading data and parity among all $N+1$ disks, rather than storing data in N disks and parity in one disk. For each

block, one of the disks stores the parity, and the others store data.

- **RAID Level 6**. RAID level 6, also called the $P + Q$ redundancy scheme, is much like RAID level 5 but stores extra redundant information to guard against multiple disk failures.

- **RAID Level 0 + 1**. RAID level 0 + 1 refers to a combination of RAID levels 0 and 1. RAID 0 provides the performance, while RAID 1 provides the reliability. Generally, this level provides better performance than RAID 5. It is common in environments where both performance and reliability are important. Unfortunately, it doubles the number of disks needed for storage, as does RAID I, so it is also more expensive.