Operating System

Unit –IV

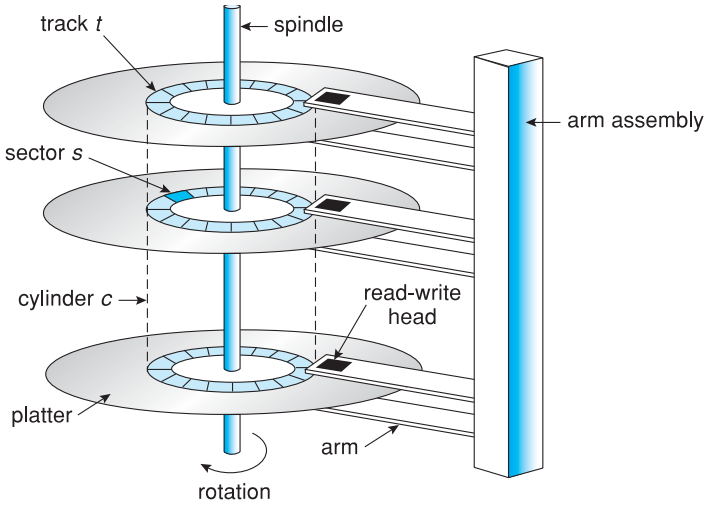
Part 1 Device Management

Overview of Mass-Storage Structure

A general overview of the physical structure of secondary and tertiary storage devices are given as follows:

Magnetic Disks

- **Magnetic disks** provide the bulk of secondary storage for modern computer systems.
- Each disk **platter** has a flat circular shape, like a CD. Common platter diameters range from 1.8 to 3.5 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**. 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. The storage capacity of common disk drives is measured in gigabytes.
- When the disk is in use, a drive motor spins it at high speed. Most drives rotate 60 to 250 times per second, specified in terms of rotations per minute **(RPM)**.
- Common drives spin at 5,400, 7,200, 10,000, and 15,000 RPM. Disk speed has two parts.
- The **transfer rate** is the rate at which data flow between the drive and the computer. The **positioning time**, or **random-access time**, consists of two parts: the time necessary to move the disk arm to the desired cylinder, called the **seek time**, and the time necessary for the desired sector to rotate to the disk head, called the **rotational latency**.



Moving Disk Head Mechanism.

Solid-State Disks

- An SSD is nonvolatile memory that is used like a hard drive. SSDs have the same characteristics as traditional hard disks but can be more reliable because they have no moving parts and faster because they have no seek time or latency.
- In addition, they consume less power. However, they are more expensive per megabyte than traditional hard disks, have less capacity than the larger hard disks, and may have shorter life spans than hard disks, so their uses are somewhat limited.
- One use for SSDs is in storage arrays, where they hold file-system metadata that require high performance. SSDs are also used in some laptop computers to make them smaller, faster, and more energy-efficient.

Magnetic Tapes

- **Magnetic tape** was used as an early secondary-storage medium. Although it is relatively permanent and can hold large quantities of data, its access time is slow compared with that of main memory and magnetic disk.
- In addition, random access to magnetic tape is about a thousand times slower than random access to magnetic disk, so tapes are not very useful for secondary storage.
- Tapes are used mainly for backup, for storage of infrequently used information, and as a medium for transferring information from one system to another.

Disk Structure

- Modern magnetic 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—at least in theory—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.

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**.

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.

Network-Attached Storage

- A network-attached storage (NAS) device is a special-purpose storage system that is accessed remotely over a data network
- 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 less efficient and have lower performance than some direct-attached storage options.

Storage-Area Network

- A storage-area network (SAN) is a private network (using storage protocols rather than networking protocols) connecting servers and storage units.
- 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.

Disk Scheduling

Note: Refer to all the examples done during the class

Disk Management

The operating system is responsible for several other aspects of disk management, too. We discuss disk initialization, booting from disk, and bad-block recovery.

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 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 ECC is an error-correcting code because it contains enough information, if only a few bits of data have been corrupted, to enable the controller to identify which bits have changed and calculate what their correct values should be. It then reports a recoverable **soft error**.
- The controller automatically does the ECC processing whenever a sector is read or written. Before it can use a disk to hold files, the operating system still needs to record its own data structures on the disk. It does so in two steps.
- The first step is to **partition** the disk into one or more groups of cylinders. The operating system can treat each partition as though it were a separate disk.
- The second step is **logical formatting**, or creation of a file system. In this step, the operating system stores the initial file-system data structures onto the disk. These data structures may include maps of free and allocated space and an initial empty directory.

Boot Block

For a computer to start running—for instance, when it is powered up or rebooted—it must have an initial program to run.

This initial **bootstrap** program tends to be simple. It initializes all aspects of the system, from CPU registers to device controllers and the contents of main memory, and then starts the operating system.

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.

For most computers, the bootstrap is stored in **read-only memory (ROM)**. This location is convenient, because ROM needs no initialization and is at a fixed location that the processor can start executing when powered up or reset.

The full bootstrap program is stored in the "boot blocks" at a fixed location on the disk. A disk that has a boot partition is called a **boot disk** or **system disk**.

Bad Blocks

- Because disks have moving parts and small tolerances (the disk head flies just above the disk surface), they are prone to failure.
- Sometimes the failure is complete; in this case, the disk needs to be replaced and its contents restored from backup media to the new disk. More frequently, one or more sectors become defective. Most disks even come from the factory with **bad blocks**.
- Depending on the disk and controller in use, these blocks are handled in a variety of ways.
- On simple disks, such as some disks with IDE controllers, bad blocks are handled manually.
- One strategy is to scan the disk to find bad blocks while the disk is being formatted. Any bad blocks that are discovered are flagged as unusable so that the file system does not allocate them. The controller maintains a list of bad blocks on the disk. The list is initialized during the low-level formatting at the factory and is updated over the life of the disk.

- Low-level formatting also sets aside spare sectors not visible to the operating system. The controller can be told to replace each bad sector logically with one of the spare sectors. This scheme is known as **sector sparing** or **forwarding**.

Swap-Space Management

- Swapping in that setting occurs when the amount of physical memory reaches a critically low point and processes are moved from memory to swap space to free available memory.
- **Swap-space management** is another low-level task of the operating system. Virtual memory uses disk space as an extension of main memory.
- Since disk access is much slower than memory access, using swap space significantly decreases system performance.
- The main goal for the design and implementation of swap space is to provide the best throughput for the virtual memory system.

Swap-Space Use

- Swap space is used in various ways by different operating systems, depending on the memory-management algorithms in use.
- For instance, systems that implement swapping may use swap space to hold an entire process image, including the code and data segments.
- Paging systems may simply store pages that have been pushed out of main memory. The amount of swap space needed on a system can therefore vary from a few megabytes of disk space to gigabytes, depending on the amount of physical memory, the amount of virtual memory it is backing, and the way in which the virtual memory is used.
- These swap spaces are usually placed on separate disks so that the load placed on the I/O system by paging and swapping can be spread over the system's I/O bandwidth.

Swap-Space Location

- A swap space can reside in one of two places: it can be carved out of the normal file system, or it can be in a separate disk partition.
- If the swap space is simply a large file within the file system, normal file-system routines can be used to create it, name it, and allocate its space. Alternatively, swap space can be created in a separate **raw partition**.
- No file system or directory structure is placed in this space. Rather, a separate swap-space storage manager is used to allocate and de-allocate the blocks from the raw partition.
- This manager uses algorithms optimized for speed rather than for storage efficiency, because swap space is accessed much more frequently than file systems.
- Some operating systems are flexible and can swap both in raw partitions and in file-system space.

RAID Structure

RAID (Redundant Array of Inexpensive Disks or Drives, or Redundant Array of Independent Disks) is a data storage virtualization technology that combines multiple physical disk drive components into one or more logical units for the purposes of data redundancy, performance improvement, or both.

RAID LEVELS

Numerous schemes to provide redundancy at lower cost by using 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.

RAID level 0. 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 0 consists of data mirroring, without parity or striping. Data is written identically to two drives, thereby producing a "mirrored set" of drives

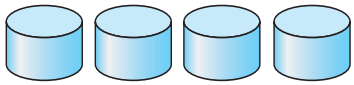
RAID level 2. RAID level 2 is also known as memory-style error-correcting code (ECC) organization. Memory systems have long detected certain errors by using parity bits. Each byte in a memory system may have a parity bit associated with it that records whether the number of bits in the byte set to 1 is even (parity = 0) or odd (parity = 1). If one of the bits in the byte is damaged (either a 1 becomes a 0, or a 0 becomes a 1), the parity of the byte changes and thus does not match the stored parity. Similarly, if the stored parity bit is damaged, it does not match the computed parity. Thus, all single-bit errors are detected by the memory system. Error-correcting schemes store two or more extra bits and can reconstruct the data if a single bit is damaged.

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. The idea is as follows: If one of the sectors is damaged, we know exactly which sector it is, and we can figure out whether any bit in the sector is a 1 or a 0 by computing the parity of the corresponding bits from sectors in the other disks. If the parity of the remaining bits is equal to the stored parity, the missing bit is 0; otherwise, it is 1.

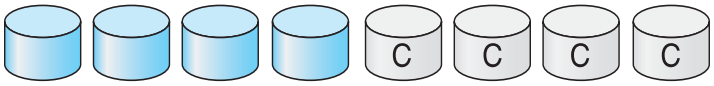
RAID level 4. RAID level 4, or block-interleaved parity organization, uses block-level striping, as in RAID 0, and in addition keeps a parity block on a separate disk for corresponding blocks from *N* other disks. If one of the disks fails, the parity block can be used with the corresponding blocks from the other disks to restore the blocks of the failed disk.

RAID level 5. RAID level 5, or block-interleaved distributed parity, differs from level 4 in that it spreads data and parity among all *N*+1disks, 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.

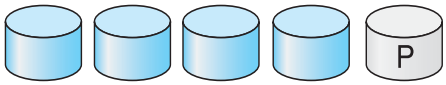


(a) RAID 0: non-redundant striping.

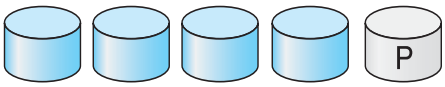


(b) RAID 1: mirrored disks.

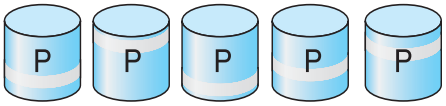
(c) RAID 2: memory-style error-correcting codes.



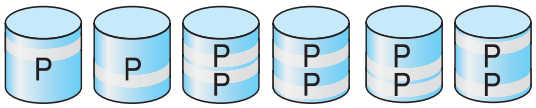
(d) RAID 3: bit-interleaved parity.



(e) RAID 4: block-interleaved parity.



(f) RAID 5: block-interleaved distributed parity.



(g) RAID 6: P + Q redundancy.