10.1 Overview of Mass-Storage Structure

This section, present a general overview of the physical structure of secondary and tertiary storage devices.

10.1.1 Magnetic Disks

Magnetic disks provide the bulk of secondary storage for modern computer systems. Conceptually, disks are relatively simple (Figure 10.1). 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.

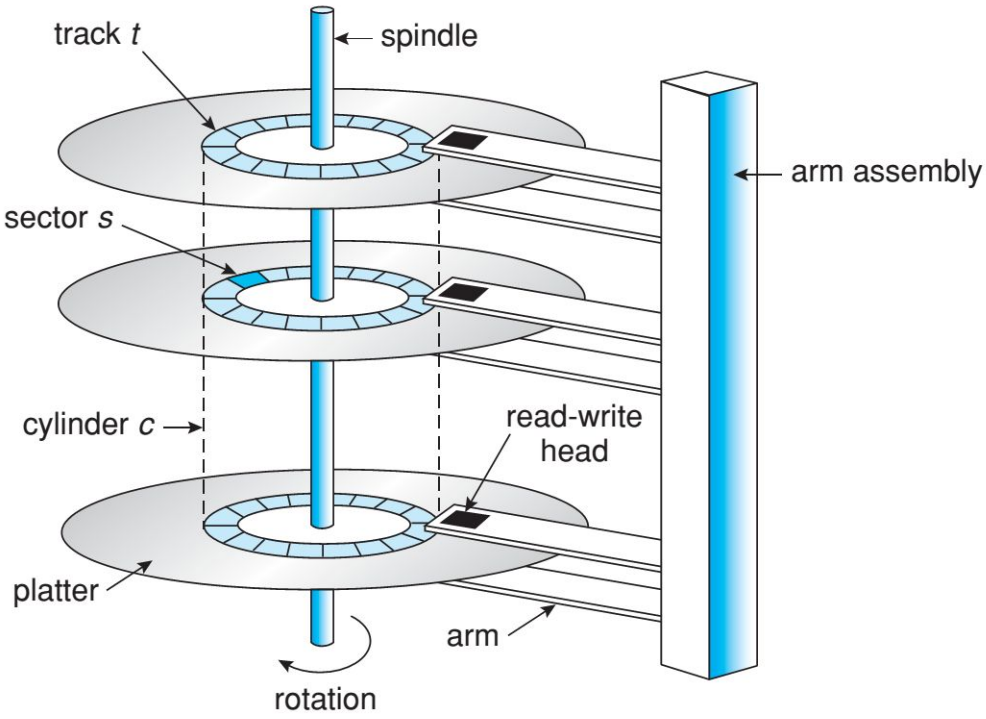

Figure 10.1 Moving-head disk mechanism.

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 15,000RPM. 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.** Typical disks can transfer several **megabytes** of data per second, and they have seek times and rotational latencies of several **milliseconds**.

Because the d**isk head flies on an extremely thin cushion of air** (measured in microns), there is a danger that the head will make contact with the disk surface. Although the disk platters are coated with a thin protective layer, the **head will sometimes damage the magnetic surface. This accident is called a head crash**. A head crash normally cannot be repaired; the entire disk must be replaced.

A disk can be **removable**, allowing different disks to be mounted as needed.

Removable magnetic disks generally consist of one platter, held in a plastic case to prevent damage while not in the disk drive. Other forms of removable disks include CDs, DVDs, and Blu-ray discs as well as removable flash-memory devices known as flash drives(which are a type of solid-state drive).

10.1 Overview of Mass-Storage Structure

` **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 advanced technology attachment (ATA), serial ATA(SATA), eSATA, universal serial bus (USB), and fibre channel (FC). 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. A disk controller is built into each disk drive.** To perform a disk I/O operation, the computer places a command into the host controller, typically using memory-mapped I/O ports, as described in Section 9.7.3. The host controller then sends the command via messages to the disk controller, and the disk controller operates the disk-drive hardware to carry out the command. **Disk controllers usually have a built-in cache**. Data transfer at the disk drive happens **between the cache and the disk surface, and data transfer to the host,** at fast electronic speeds, occurs between the cache and the host controller.

10.1.2 Solid-State Disks

Sometimes old technologies are used in new ways as economics change or the technologies evolve. An example is the growing importance of **solid-state Disks**,or SSDs. Simply described, an **SSD is nonvolatile memory that is used like a hard drive.** *There are many variations of this technology, from DRAM with a battery to allow it to maintain its state in a power failure through flash-memory technologies like single-level cell (SLC) and multilevel cell (MLC) chips.* **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**.

Because SSDs can be much faster than magnetic disk drives, standard bus interfaces can cause a major limit on throughput. Some SSDs are designed to connect directly to the system bus (PCI, for example). *SSDs are changing other traditional aspects of computer design as well. Some systems use them as a direct replacement for disk drives, while others use them as a new cache tier, moving data between magnetic disks, SSDs, and memory to optimize performance.*

10.1.3 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.

A tape is kept in a spool and is wound or rewound past a read–write head.

Moving to the correct spot on a tape can take minutes, but once positioned, tape drives can write data at speeds comparable to disk drives.

Tape capacities vary greatly, depending on the particular kind of tape drive, with current capacities exceeding several terabytes. Some tapes have built-in compression that can more than double the effective storage. Tapes and their drivers are usually categorized by width, including 4, 8, and 19 millimeters and 1/4 and 1/2 inch. Some are named according to technology, such as LTO-5 and SDLT.

10.2 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**. This option is described in Section 10.5.1. 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.

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. **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**. This method is used in CD-ROM And DVD-ROM drives. Alternatively, the disk rotation speed can stay constant; in this case, the **density of bits decreases from inner tracks to outer tracks to keep the data rate constant.**

10.3 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.

10.3.1 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 **fibre channel (FC), a high-speed serial architecture that can operate over optical fiber or over a four-conductor copper cable.**

10.3.2 Network-Attached Storage

A network-attached storage (NAS) **device is a special-purpose storage system that is accessed remotely over a data network** (Figure 10.2). 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.

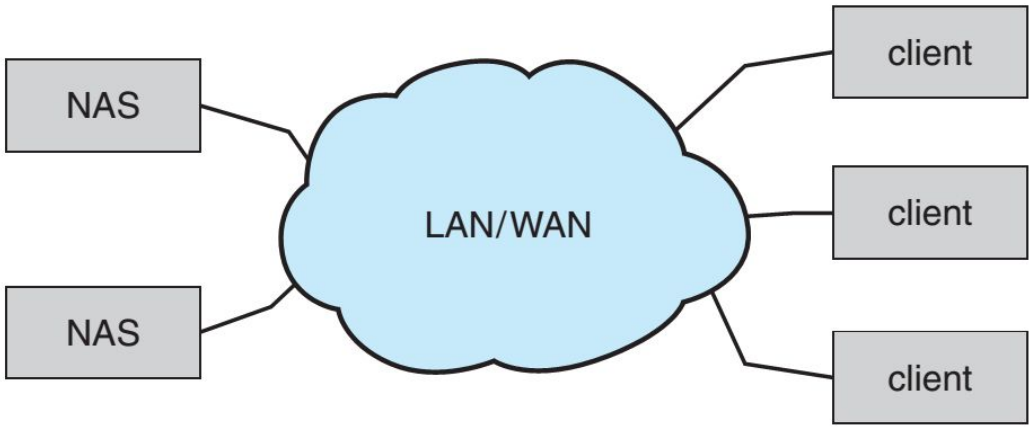

Figure 10.2 Network-attached storage.

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.**

One **drawback** of network-attached storage systems is that **the storage I/O operations consume bandwidth on the data network,** thereby increasing the latency of network communication. This problem can be particularly acute in large client –server installations.

10.4 Disk Scheduling

One of the responsibilities of the operating system is to use the hardware efficiently.

For the disk drives, meeting this responsibility entails having **fast access time and large disk bandwidth.** For magnetic disks, the access time has two major components, as mentioned in Section 10.1.1. **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 managing the order in which disk I/O requests are serviced. Whenever a process needs I/O to or from the disk, it issues a system call to the operating system. The request specifies several pieces of information:

•Whether this operation is input or output
•What the disk address for the transfer is
•What the memory address for the transfer is
•What the number of sectors to be transferred is

If the desired disk drive and controller are **available**, the request can be serviced immediately. If the drive or controller is **busy**, any new requests for service will be placed in the queue of pending requests for that drive. For a **multiprogramming** system with many processes, the disk queue may often have several pending requests. Thus,when one request is completed, the operating system chooses which pending request to service next. How does the operating system make this choice? Any one of **several disk-scheduling algorithms** can be used.

10.5 Disk Management

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

10.5.1 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 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 (Section 10.5.3). The **ECCis 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.

Most hard disks are low-level-formatted at the factory as a part of the manufacturing process. This **formatting enables the manufacturer to test the disk and to initialize the mapping from logical block numbers to defect-free sectors on the disk.** For many hard disks, when the disk controller is instructed to low-level-format the disk, it can also be told how many bytes of data space to leave between the header and trailer of all sectors. **It is usually possible to choose among a few sizes, such as 256, 512, and 1,024 bytes.** Formatting a disk with a larger sector size means that fewer sectors can fit on each track; but it also means that fewer headers and trailers are written on each track and more space is available for user data. Some operating systems can handle only a sector size of 512 bytes.

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. For instance, one partition can hold a copy of the operating system's executable code, while another holds user files.
- 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.

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.**

Some operating systems give special programs the ability to use a disk partition as a large sequential array of logical blocks, without any file-system data structures. **This array is sometimes called the raw disk ,and I/O to this array is termed raw I/O.** For example, some database systems prefer raw I/O because it enables them to control the exact disk location where each database record is stored. *Raw I/O bypasses all the file-system services, such as the buffer cache, file locking, prefetching, space allocation, file names, and directories. We can make certain applications more efficient by allowing them to implement their own special-purpose storage services on a raw partition, but most applications perform better when they use the regular file-system services.*

10.5.2 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, t**he 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. And, since ROM is read only, it cannot be infected by a computer virus. The problem is that changing this bootstrap code requires changing the ROM hardware chips.

For this reason, most systems store a tiny bootstrap loader program in the boot ROM whose only job is to bring in a full bootstrap program from disk. The full bootstrap program can be changed easily: a new version is simply written onto the disk. T**he 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.**

The code in the boot ROM instructs the disk controller to **read the boot blocks into memory** (no device drivers are loaded at this point) and then starts **executing that code.** The full bootstrap program is more sophisticated than the bootstrap loader in the boot ROM. **It is able to load the entire operating system from a non-fixed location on disk and to start the operating system running.** Even so, the full bootstrap code may be small.

Let's consider as an example the boot process in Windows. First, note that Windows allows a hard disk to be divided into partitions, and one partition identified as the boot partition—contains the operating system and device drivers. The **Windows system places its boot code in the first sector on the hard disk**, which it terms the **master boot record,or MBR**. Booting begins by running code that is resident in the system's ROM memory. **This code directs the system to read the boot code from the MBR.** In addition to containing boot code, the **MBR contains a table listing the partitions for the hard disk and a flag indicating which partition the system is to be booted from,** as illustrated in Figure 10.9.

Once the system identifies the boot partition, it reads the first sector from that partition (which is called the boot sector) and continues with the remainder of the boot process, which includes loading the various subsystems and system services.

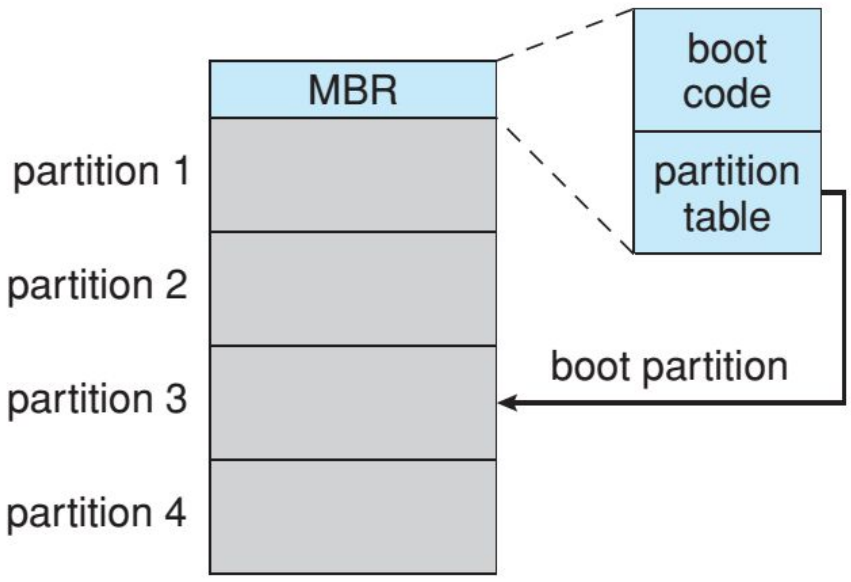

Figure 10.9 Booting from disk in Windows.

10.5.3 Bad Blocks

Because disks have moving parts and small tolerances (recall that 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. If blocks go bad during normal operation, a special program (such as the Linux bad blocks command) **must be run manually to search for the bad blocks and to lock them away.** Data that resided on the bad blocks usually are lost.
- More **sophisticated disks are smarter about bad-block recovery**.

- 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.

A typical bad-sector transaction might be as follows:

•The operating system tries to read logical block 87.

•The controller calculates the ECCand finds that the sector is bad. It reports this finding to the operating system.

•The next time the system is rebooted, a special command is run to tell the controller to replace the bad sector with a spare.

•After that, whenever the system requests logical block 87, the request is translated into the replacement sector's address by the controller.

Note that such a redirection by the controller could invalidate any optimization by the operating system's disk-scheduling algorithm! For this reason, most disks are formatted to provide a few spare sectors in each cylinder and a spare cylinder as well. When a bad block is remapped, the controller uses a spare sector from the same cylinder, if possible.

As an alternative to sector sparing, some controllers can be instructed to replace a bad block by sector slipping. Here is an example: Suppose that *logical block 17 becomes defective and the first available spare follows sector 202. Sector slipping then remaps all the sectors from 17 to 202, moving them all down one spot. That is, sector 202 is copied into the spare, then sector 201 into 202, then 200 into 201, and so on, until sector 18 is copied into sector 19. Slipping the sectors in this way frees up the space of sector 18 so that sector 17 can be mapped to it.*

The replacement of a bad block generally is not totally automatic, because the data in the bad block are usually lost. **Soft errors may trigger a process in which a copy of the block data is made and the block is spared or slipped. An unrecoverable hard error, however, results in lost data. Whatever file was using that block must be repaired (for instance, by restoration from a backup tape), and that requires manual intervention.**