Hard Disk Organization

Vocabulary

Platter: one ceramic plate, covered with magnetizable film where the bits are actually stored. Both sides of a platter can be used. Increasing the number of platters is one way to increase capacity. Another way is to increase the density of bits stored on a platter.

Spindle: axis on which multiple platters are mounted in a stacked arrangement. When the disk is in operation, all platters spin in sync around the spindle at an RPM defined by the disk spec. Unlike old floppy disks, a hard disk is spinning at a constant speed at all times when in operation and not in sleep mode. As a matter of manufacturing practice, common speeds are 5400, 7200, and 10000 RPM.

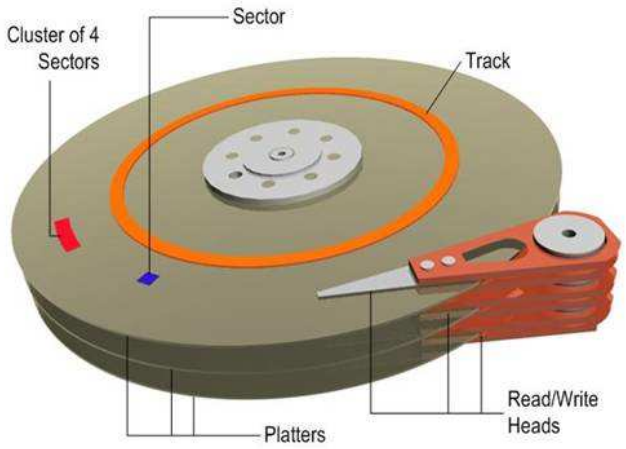
Read Write Heads: sensor at the end of an arm that hovers over the magnetized surface of the platter without ever touching it. There is a r/w head for every platter surface (both top and bottom). The heads all move in unison, but only one sensor is active at a time. Micro-alignment issues for bit positions on different surfaces will prevent multiple surfaces from being accessed at the same time.

Track: each surface of each platter is divided into concentric rings called tracks. Bits are stored along the ring defined by the track. A typical disk today will have tens of thousands of tracks. Contrast this to an LP or CD which has a single spiral track rather than multiple concentric tracks. As the disk spins, if the r/w heads do not move, a single track will pass underneath the r/w head. The r/w heads must be moved to move to a different track. Two important performance measures related to the positions of the r/w head are (1) rotational latency and (2) seek time. Note that spin rate affects performance, not capacity.

Cylinder: all tracks at a given radius from the spindle across all platters and heads. When storing info it is useful from a performance perspective for related data to be stored in a single cylinder since it can be accessed without having to move the read-write heads (no seek).

Rotational Latency: the time required for a stationary r/w head to wait for information on the current track but on the other side of the disk to pass by underneath as the disk spins. This figure is closely related to the speed at which the disk spins.

Seek time: the time required to move the r/w head from its current track to another track to access information stored there. This is not related to the speed at which the disk spins.



Sector: also called a **disk block**, a series of consecutive bits along a track. Every sector has a unique logical disk address called CHS: cylinder-head-sector. A sector is the smallest **addressable** unit of storage on the disk. A common size is **512 bytes**. Some older documentation uses the term "sector" to refer to all sectors on all tracks within a specific pie shaped wedge. In this terminology, the disk block is the intersection of a track and a sector.

Cluster: a small group of consecutive sectors. The cluster is the smallest unit of **allocatable** storage on the disk. This will be significant when we talk about file systems shortly. When a file is created, space for the file is allocated one sector at a time, which can result in wasted space. For example, suppose a file requires 7 sectors of storage, but clusters are 4 sectors each. The file will have to be allocated two clusters, and there will be 1 sector of wasted space in the second cluster that cannot be allocated to another file.

Zone Recording

In older disks, all tracks were divided into the same number of sectors. The advantage is that it is simple, but the disadvantage is that tracks near the center store bits more densely, while tracks near the outside edge store them more sparsely, resulting in wasted capacity.

Zone recording allows tracks near the outside edge to have more sectors than tracks near the interior. See figure. Some group of tracks with the same number of sectors form a single zone and all tracks in the same zone have the same number off sectors. There can be more than one track per zone.

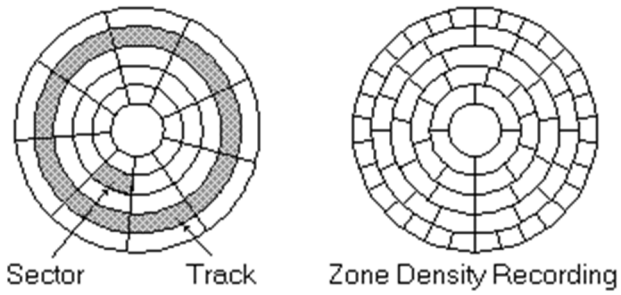


Figure taken from
http://www.dewassoc.com/kbase/hard_drives/hard_disk_sector_structures.htm

Controller: the hardware chip inside the disk housing that acts as the interface between the OS and the content of the disk. In older systems, the OS had to keep track of CHS addresses. In newer systems, zone recording makes CHS addressing much more complex, have switched to logical block addresses (LBA). LBA just treats the disk as a huge linear array of blocks or sectors numbered from 0 to n-1, where n is the total number of blocks (a very big number). The OS uses LBA addressing, the disk controller translates LBA to CHS. CHS is based on physical geometry of the disk, LBA is based on logical geometry.

On-Disk Structures

A newly manufactured hard drive needs 3 formatting steps before it is ready to use by a computer platform and its operation system: (1) low level formatting, (2) partitioning, and (3) high level formatting.

Low Level Formatting

The disk controller is the interface between the operating system and the information on the disk. It organizes info into blocks or sectors, each with its own unique address of (surface, track, sector). Low level formatting establishes the physical location of tracks and sectors and records the info in tables used by the disk controller. These days the end user cannot usually change this information.

Partitioning

The next step in making the space on the disk accessible and usable is to create one or more partitions. A partition is a logical divisionof the physical hard disk drive that can be used by the operating system as if it were a separate drive. The simplest partitioning step is to treat the entire disk as one partition, but very large disks can be divided into multiple partitions for purposes such as dual boot – one OS and file system installed in one partition, a different OS and file system in another. From the point of view of the OS, each partition looks like a separate drive, even though they might be on the same physical disk.

At a minimum, partitioning a hard disk requires the creation of a global table of contents structure traditionally called the MBR or master boot record in the IBM PC environment. The MBR contains a table of contents that points to the beginning of each defined partition. If there are multiple bootable partitions, the MBR may contain the code for the boot loader that runs on system startup and allows the user to pick which partition to complete the boot from. If there is only one partition, the MBR will contain the code for the first stage boot loader for the OS in that partition.

High Level Formatting

Within each partition the end user can perform high level formatting to create a file system and install a bootable OS. Creation of a file system requires the clusters in the partition to be organized into a free list plus a root directory. Exact details depend on which specific file system is being installed. Popular choices are FAT (older) and NTFS (newer) for MS Windows, and ext4 for Linux. Once a file system has been created, the partition can be made bootable by installing an OS into that file system. Different partitions can support different OSes and different file systems. A non-bootable partition can be used for storage only.

The Boot Process

Boot refers to the process of taking a computer system from its powered down state, powering up, loading the OS into RAM, launching it, and having it take control. When a system is first powered on, all volatile memory including RAM is empty. Boot takes place in several steps. The process described below is based on the IBM PC model, other platforms will have equivalents but named differently.

- **POST**: power on self test, confirms that basic low level hardware is installed and operational: RAM, keyboard, mouse, display.
- **BIOS**: basic I/O system, a simple OS that runs from ROM. Newer versions of BIOS are called Unified Extensible Firmware Interface (UEFI). BIOS does little more than load the boot loader into RAM and turn control over to it.
- **Boot Loader**: provides the user a simple menu on the display to allow the user to pick from available bootable partitions. This step may be skipped if the hard drive was formatted with only a single bootable partition.
- First (and possibly second stage) boot: large OSes like MS Windows cannot be loaded into RAM in a single step. Boot is divided into stages. First stage is a reduced functionality version of the OS consisting of simply the kernel and key generic device drivers. Second stage loads additional modules and other drivers.
- Complete OS: finally the full OS is loaded into memory and it assumes control of the system.

Some Popular File Systems

FAT: **File Allocation Table** FS, popular for early versions of MS Windows, still used on floppy disks. Worth studying because of its simplicity and reliability. The FAT FS maintains two important items:

- the file system itself (files and directories)
- the FAT table.

There are two kinds of file: regular files that contain data, and directories or folders that contain only references to other files (which can include more directories). The root of the file system is a special directory called slash "\" and is placed at a specific location near the beginning of the disk so that it can be found during boot. Starting at the root directory, any file in the file system can be located by moving downward through directories and subdirectories, following pointers until the desired file is reached. A parent directory for a collection of files contains an entry for each of its children. This entry includes the id of the first cluster in that file, which is used to physically locate the file on the disk. If the file extends over multiple clusters, the FAT table must be consulted to find the id of the next cluster. The FAT table is a small table that holds status information about each cluster in the partition, indicating if the cluster is (1) part of a file with a continuation cluster following, (2) part of a file and the last cluster in the file, or (2) unused/free. To summarize, the FAT FS organizes its files and directories by means of two data structures:

- Directory entries (lots of info about a file stored here)
- The FAT table (a single item of info about a cluster stored here)

Most of the information about a file including its starting cluster is stored in the directory. Directory entries provide info about files. The FAT table holds a small but critical info about each individual cluster in the partition. The FAT table provides info about clusters. As you're moving through the directory tree, you only need to check the FAT table to find out if the current file fits within the current cluster or continues on into another cluster. If all files fit in a single cluster, there would be no need for the FAT table.

NTFS: Current FS for MS Windows. Not open source, detailed knowledge of its structure has been assembled through reverse engineering efforts. Every file in the file system has an entry in the **MFT** (master file table). The MFT is basically the replacement for the FAT table in the FAT FS. Information about files in the MFT is divided up into **attributes**. Resident attributes are attributes that fit inside the MFT entry. Some attributes such as the content of the file may be too big to fit in the entry directly and spill out into the clusters section of the disk, these are nonresident attributes. All but very small directory files have their entries stored as nonresident attributes, organized as B trees for faster lookup of files in very large directories.

ReFS: Resiliant FS, Microsoft's replacement for NTFS. Currently only for file servers, not a bootable FS, this could change any day now. B trees are now used for all on disk structures, not just directories.

Ext4: extended file system version 4, current file system for Linux. Ext4 follows previous versions ext2 and ext3. Groups of sectors are called blocks rather than sectors. Files are spread across allocated blocks and the blocks are maintained in a tree-like data structure called an inode. Every file must have its own inode. Since blocks and inodes are managed separately, it is possible to run out of inodes even if there are blocks available. This could happen in the case of a very large number of very small files.

Improvements have been introduced in version 4 to speed up the performance of allocating space for very large files. For example, rather than a loop that allocates one block at a time for thousands of blocks, a variable-sized **extent** can be allocated to create space for a large file in one operation.

YAFFS: Yet Another Flash File System, used in solid state memory. Google uses it in many Android devices.