CI583: Data Structures and Operating Systems
File systems: improvements over the early systems

Outline

Media

So far, we have been ignoring the issue of the actual media that data blocks are stored on.

We have been assuming that any block of data can be retrieved with the same cost (O(1)), as if we were accessing locations in a big array.

For a solid-state drive (SSD) this is more or less true.

Media

However, if our data is stored on a (mechanical) hard disk drive (HDD) this is very far from being the case. The fact that S5FS also makes this simplifying assumption is one of the reasons for its poor performance.

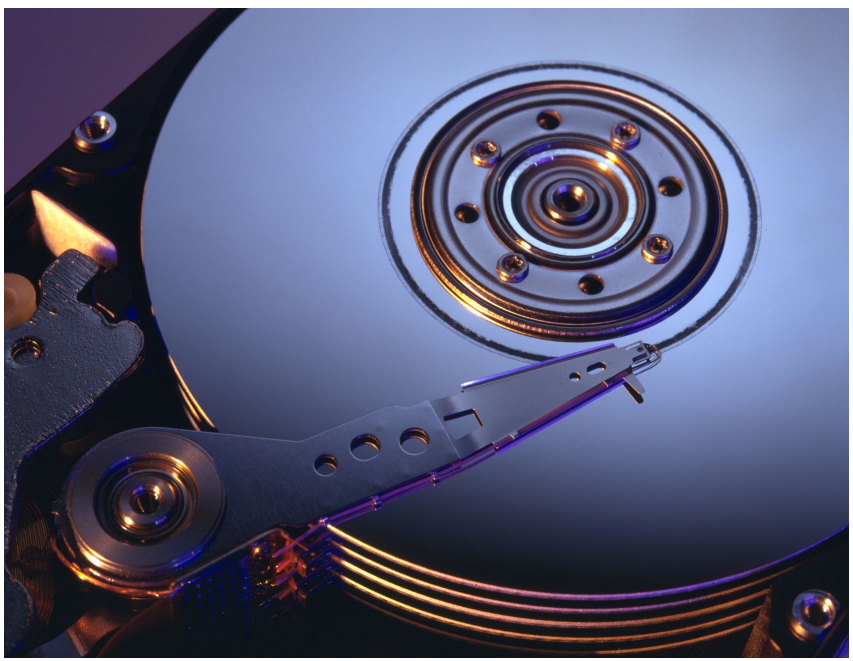


Image copyright `http://www.file-recovery.com/`

Disk architecture

A typical disk drive consists of several platters each of which has one or two recording surfaces.

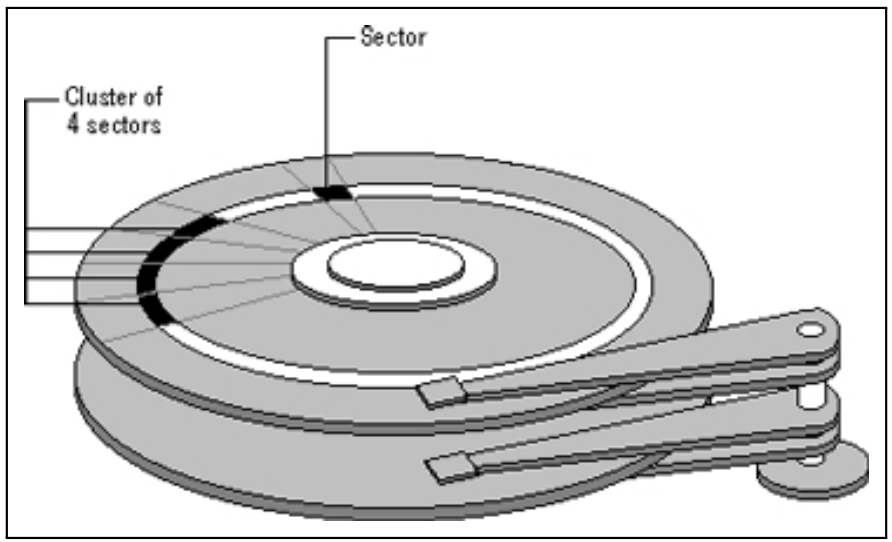
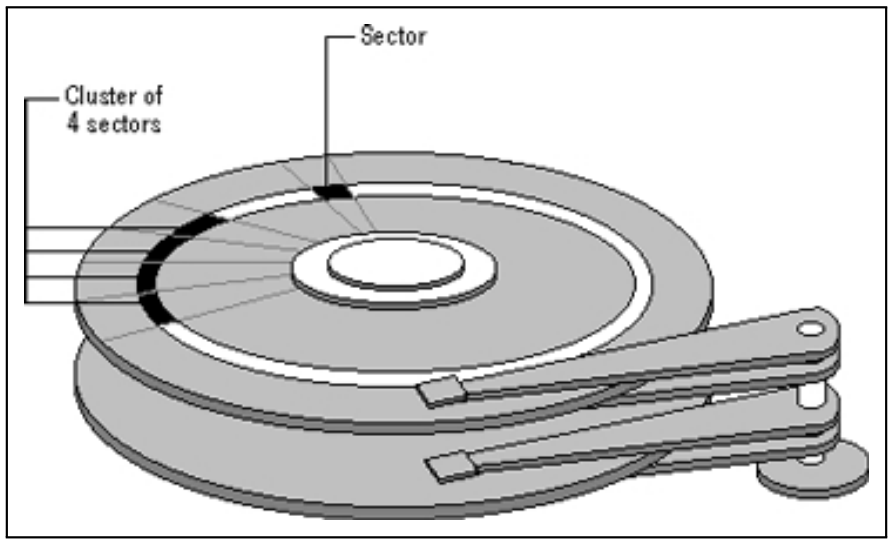


Image copyright `http://www.file-recovery.com/`

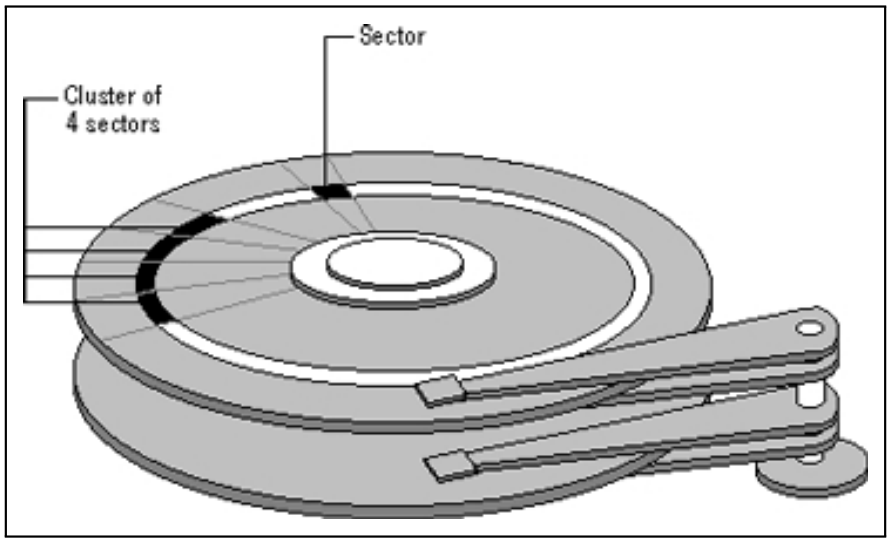
Disk architecture

Each surface is divided into a number of concentric tracks, and each track is divided into a number of sectors. All the sectors are the same size, and outer tracks have more sectors.



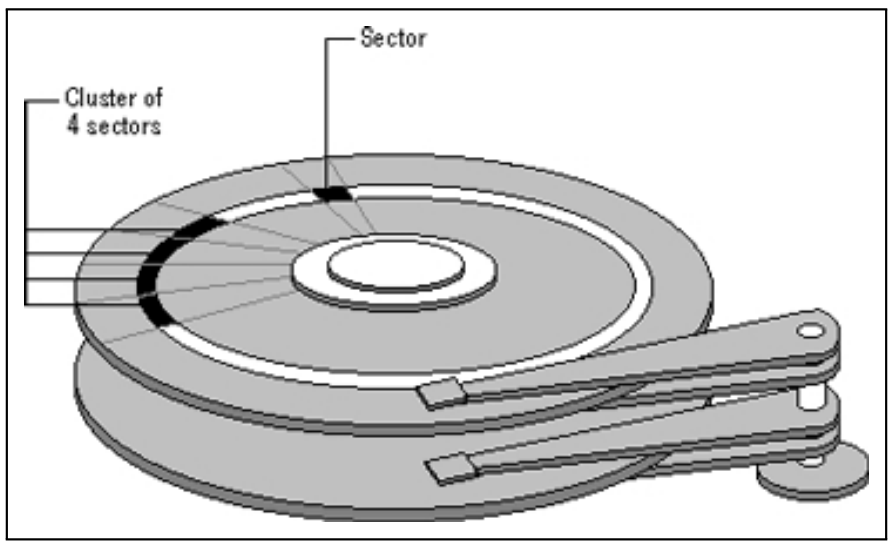
Disk architecture

Data is read and written using a set of read/write heads, one per surface. The heads are connected to arms that move together across the surfaces. Only one head is active at any one time. The set of tracks that is under the heads at any one time is called a cylinder.



Disk architecture

It should be clear that the idea of the cylinder is an important one – if we take steps to store data on separate surfaces but within the same cylinder we can switch from one head to another (which we can assume takes no time at all) without needing to spin the surfaces. To accommodate this, the position of the heads is offset from each other.



Disk architecture

In order to read or write to a location on disk, the disk controller does the following:

1. Move the heads over the correct cylinder. This is the seek time.
2. Rotate the platter until the desired sector is under the head. This is the rotational latency.
3. Rotate the platter so that the entire sector passes under the head, in order to read or write data. This is the transfer time.

Disk architecture

Rotational latency depends on the rate at which the disk spins (e.g., 10,000 RPM), and transfer time depends on the spin rate and the number of sectors per track.

Average seek time is usually the most important factor, in the low milliseconds in a typical drive – a *very* long time compared to the speed at which processors work.

An efficient file system has to take steps to reduce this.

Optimisations

With regard to storage media, the two key optimisations the designer of a file system can make are to reduce seek time and increase the amount of data transferred.

A straightforward way to reduce seek time is to use buffering.

Optimisations

Just as the OS buffers data from the file system (fetches more than it needs and stores recently accessed data in a cache), disk controllers use a pre-fetch buffer in which the whole of the most recently accessed sector is stored.

This will improve latency for reads but not writes.

Optimisations

With regard to storage media, the two key optimisations the designer of a file system can make are to reduce seek time and increase the amount of data transferred.

Other improvements (used in file systems such as Linux `ext2`):

1. Increased block size. Helpful, but complex data allocation strategies are need to avoid excessive fragmentation (see Doeppner, section 6.1).
2. Reduce seek time by data allocation strategies. Allocate the next block in a way that takes disk architecture into account – use as few cylinders as possible.

Optimisations

3. Reduce rotational latency by data allocation strategies.
   Allocate blocks so that they can be read without repositioning the heads.

   This is not a simple as allocating the blocks sequentially – the OS reads data one block at a time. So, to access two blocks, the OS issues the first instruction then waits for an interrupt to say that the work is complete.

   By the time the OS responds by asking for the next block, the disk has spun past the subsequent block and will need to complete an entire revolution before the heads are over it again. So, subsequent blocks are interleaved to give the OS time to ask for them just as they are approaching the heads.

Optimisations

4. Clustering. Allocate blocks in groups, rather than one by one. `ext2` pre-allocates 8 blocks at a time, eventually giving them up if they are not used or space becomes short.
5. Aggressive caching. If memory is abundant, maintain a very large cache and pre-fetch entire files. Works well for reading. For writing, we need to periodically write the cached data back to disk and maintain a log of updates so that work is not lost in case of a crash.