Secondary Storage

Any modern computer system will incorporate (at least) two levels of storage:

primary storage:	**random access memory (RAM)**
typical capacity	256MB to 4GB
cost per MB	$0.10
typical access time	5ns to 60ns

secondary storage:	**magnetic disk/optical devices/tape systems**
typical capacity	20GB to 400GB for fixed media; ∞ for removable
cost per MB	$0.001 for fixed media, more for removable
typical access time	8ms to 12ms for fixed media, larger for removable

Note: all statistics here are guaranteed invalid by Oct 15, 2005.

Units of Measurement

Spatial units:

byte (B)	8 bits
kibibyte (KiB)	1024 or 2^{10} bytes
mebibyte (MiB)	1024 kibibytes or 2^{20} bytes
gibibyte (GiB)	1024 mebibytes or 2^{30} bytes

IEC standard

byte (B)	8 bits
kilobyte (KB)	1024 or 2^{10} bytes
megabyte (MB)	1024 kilobytes or 2^{20} bytes
gigabyte (GB)	1024 megabytes or 2^{30} bytes

traditional

byte (B)	8 bits
kilobyte (KB)	1000 or 10^3 bytes
megabyte (MB)	1000 kilobytes or 10^6 bytes
gigabyte (GB)	1000 megabytes or 10^9 bytes

alt. industry

Time units:

nanosecond (ns)	one-billionth (10^{-9}) of a second
microsecond (μs)	one-millionth (10^{-6}) of a second
millisecond (ms)	one-thousandth (10^{-3}) of a second

Primary versus Secondary Storage

While the particular values given earlier are volatile, the relative performances suggested are actually quite stable:

Primary storage:

- costs several hundred times as much per unit as secondary storage.
- has access times that are 250,000 to 1,000,000 times faster than secondary storage
- has transfer rates that are ?? times faster than secondary storage

Why do WE care (in a data structures class)?

Often data must be first read from disk into memory for processing, and then results must be written back to disk after processing.

In many cases, data sets are too large to store in memory at once.

File Storage

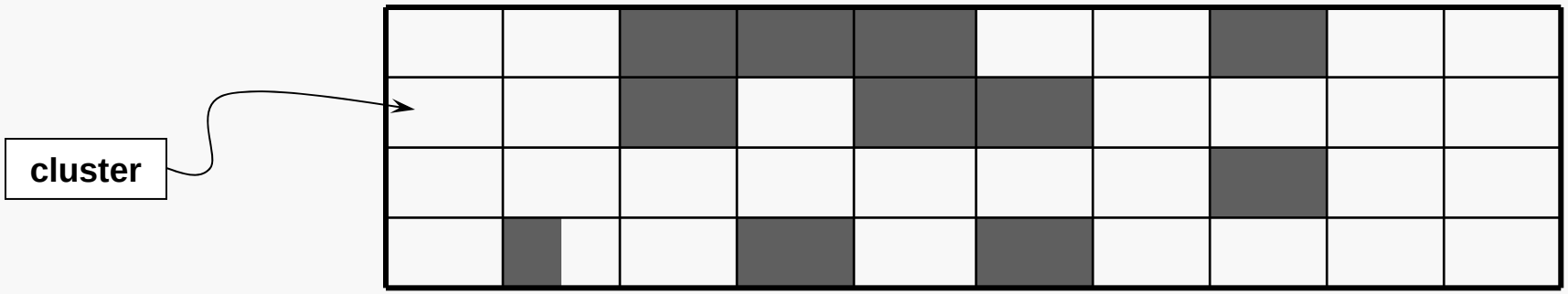
In many file systems, space is allocated in fixed-size chunks called clusters.

Typical cluster sizes range from 0.5KB up to 32KB, usually equaling a power of 2.

When a file is stored on disk, an integer number of clusters are allocated for the file; since files sizes are typically not a multiple of the cluster size, this means that a certain amount of space is wasted to internal fragmentation.

On average, ½ of a cluster is wasted per file stored. Obviously that adds up… but that's not really our concern in this course.

The clusters are also not usually stored contiguously on the disk. This external fragmentation can cause a serious degradation of performance when a file is being read from or written to disk.



Hard Disk Performance Factors

To illustrate the issues, we will consider the physical organization and behavior of a typical hard disk design.

Note that the presentation here is an over-simplification and that contemporary hard drive designs incorporate control sophistication not discussed here.

Despite advances, the basic performance issues remain the same.

We will consider three primary factors that affect the time required to read requested data from the disk into memory:

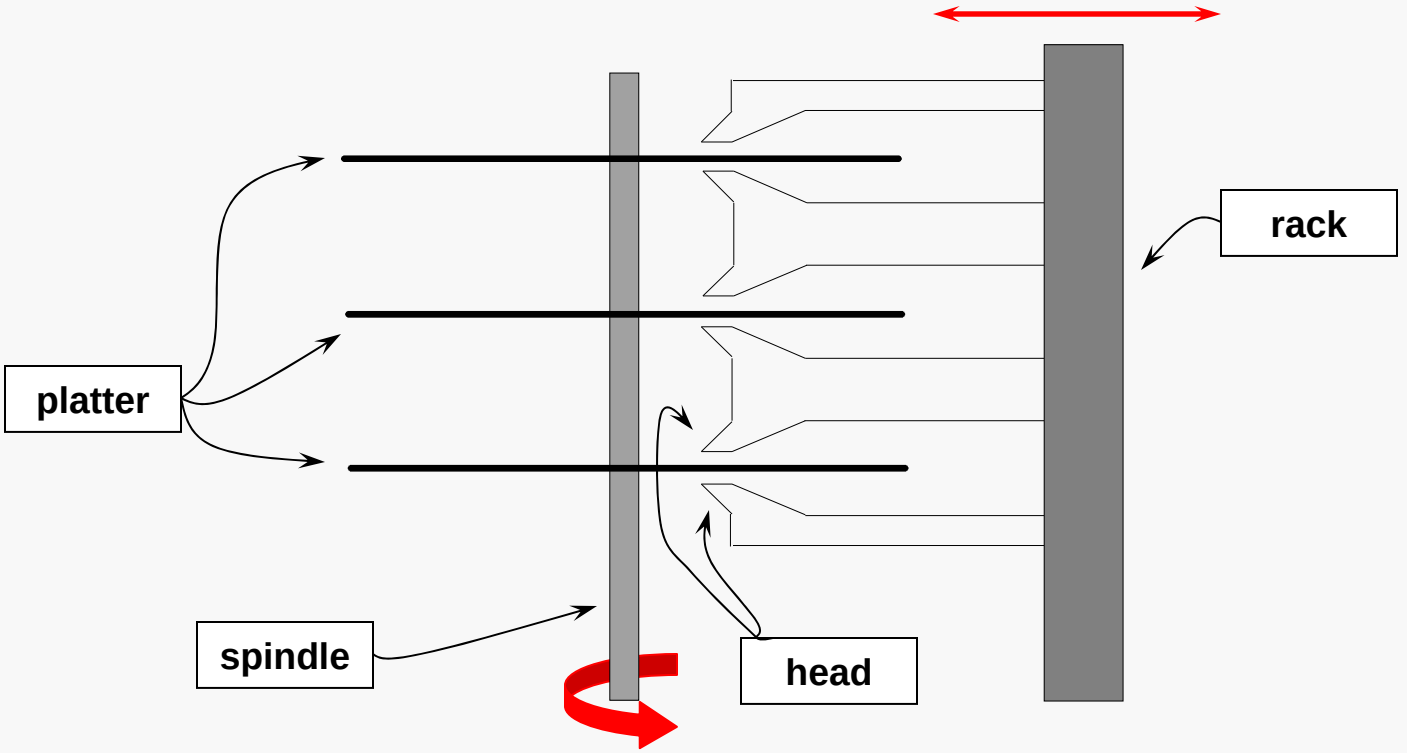
seek time	time for the appropriate I/O head to reach the desired track
latency	time for the appropriate sector to rotate to the I/O head
transfer time	time for the data to be read to move past the I/O head

First we must have a clear picture of the hardware…

Platters and I/O Heads

A typical hard drive contains a number of circular platters, attached to a rotating spindle. Each platter holds data on one or both of its surfaces.

The data is read and written by I/O heads, typically one per data surface. These I/O heads are mounted in a rack and all move in and out in unison. Typically, only one I/O head can be active (reading or writing data) at any given time.

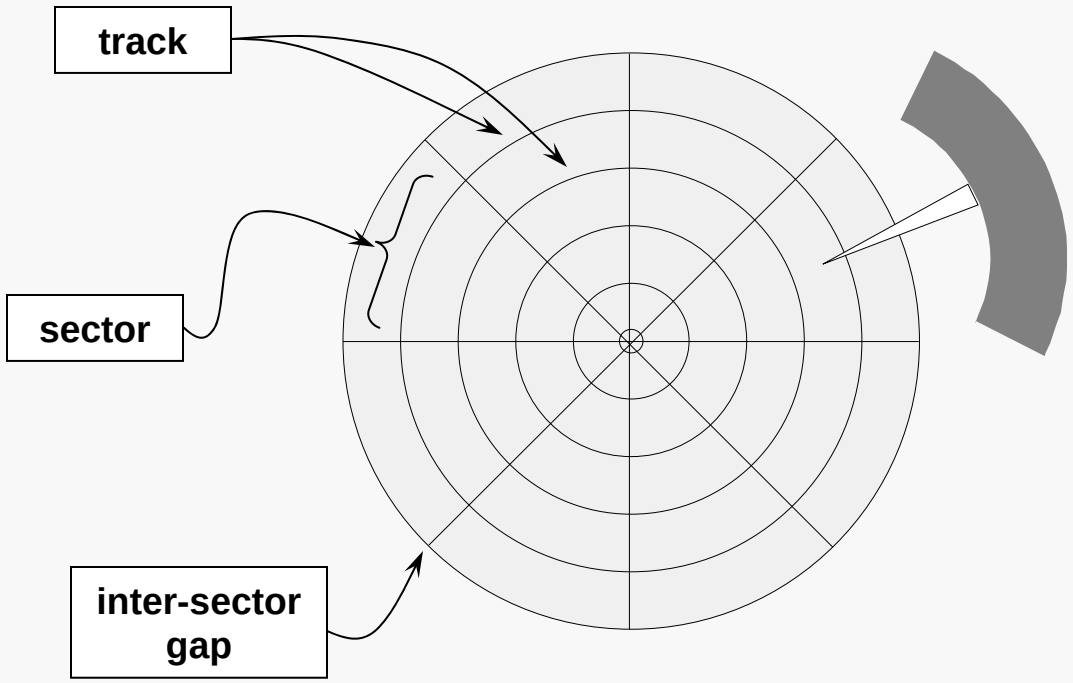


Tracks and Sectors

Each data surface is organized into concentric rings of data, called *tracks*.

Each track is divided into a number of segments, called *sectors*. On modern drives, outer tracks contain more sectors than inner tracks. We will simplify things by considering each track to hold the average number of sectors.

Each sector contains the same amount of data.



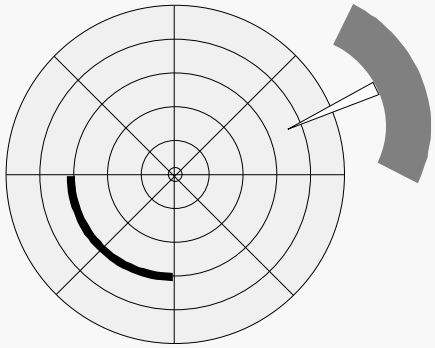
The basic unit of disk I/O is the sector.

We will see shortly that reading an entire sector from disk takes only slightly longer than reading a single byte.

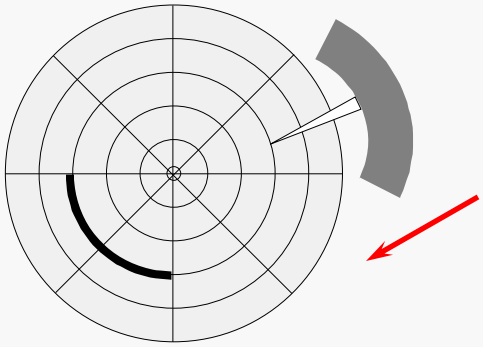
Reading a Sector

When a disk read is requested the following actions must be carried out:

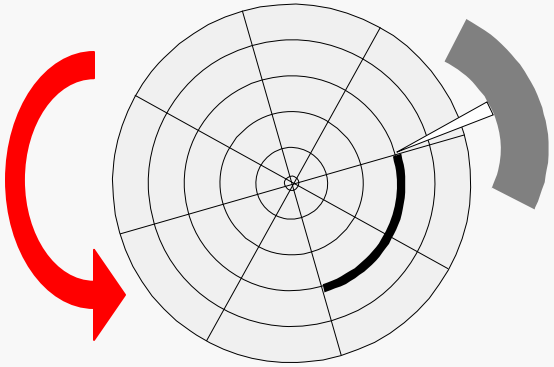
1. **Determine what sector(s) must be read — the disk controller is responsible for this.**



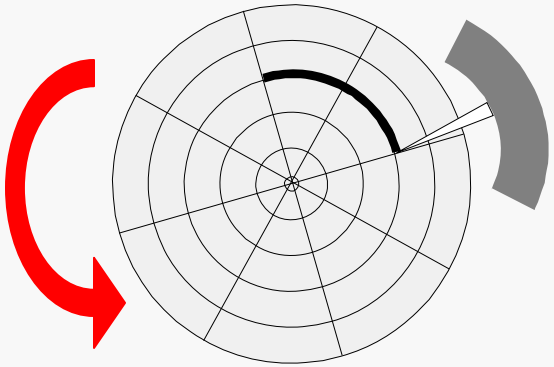
2. **Move the read head to the track containing the targeted sector(s).**



3. **Wait for the beginning of the first sector to rotate to the head.**



4. **Read the data as it rotates beneath the head.**



Seek Time

seek time — the time required to move the head to the appropriate track

The seek time depends only on the speed with which the head rack moves, and the number of tracks that the head must move across to reach its target.

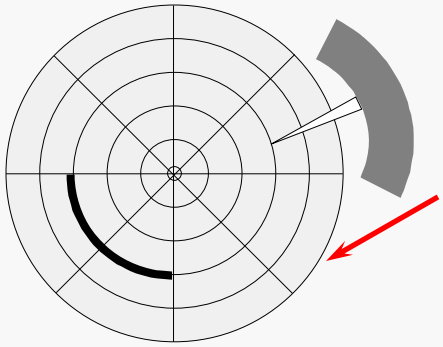
Given the following (which are constant for a particular disk):

H_s = the time for the I/O head to start moving

H_T = the time for the I/O head to move from one track to the next

then the time for the head to move n tracks is:

$$\text{Seek}(n) = H_s + H_T n$$



QTP: On average, the head will move 1/3 of the way across the platter. Why?

Rotational Latency Time

latency — the time required for the appropriate sector to rotate to the position of the I/O head

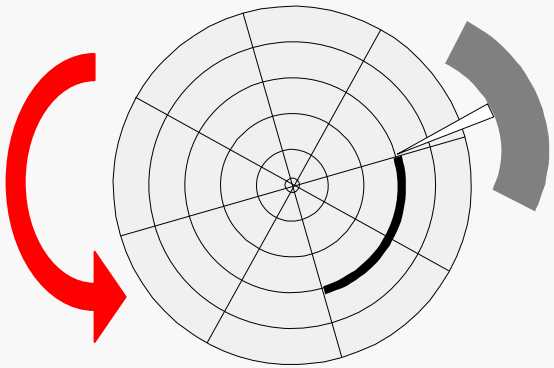
The rotational latency time depends only on the speed at which the spindle rotates, and the angle through which the track must rotate to reach the I/O head.

Given the following:

R = the rotational speed of the spindle (in rotations per minute)

θ = the number of radians through which the track must rotate

then the rotational latency (in ms) is:



$$\text{Latency}(\theta) = \left(\frac{\theta}{2\pi}\right) \times \left(\frac{60000}{R}\right)$$

On average, the latency time will be the time required for the platter to complete 1/2 of a full rotation.

Data Transfer Time

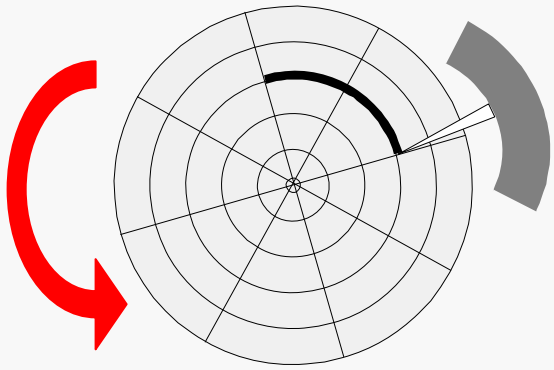
transfer time — the time required for the appropriate sector(s) to rotate under the I/O head (for contiguous sectors on the same track)

The transfer time depends only on the speed at which the spindle rotates, and the number of sectors that must be read.

Given:

S_T = the total number of sectors per track

the transfer time for n contiguous sectors on the same track is:



$$\text{Transfer}(n) = \left(\frac{n}{S_T}\right) \times \left(\frac{60000}{R}\right)$$

For sectors on different tracks, each track must be analyzed separately, allowing for seek and latency.

Total Read/Write Time

The total time to read/write data is from/to disk is then the sum of the seek time, the rotational latency time, and the transfer time.

This ignores the time for controller logic, delays due to multitasking queues, etc.; that is fair because those times are normally orders of magnitude less that the times considered.

One note: we have assumed in the analysis of the data transfer time that the I/O heads are capable of reading/writing data as fast as the sector moves beneath the head. On older disk systems that was often not the case, and interleaving was necessary to optimize read/write performance. On most contemporary disk systems, the I/O heads are fast enough that such tricks are unnecessary.

Example: Contiguous File Read

Assume a disk system with the following parameters:

# of tracks per surface	8096
avg# of sectors per track	2048
sector size	0.5 KB
cluster size	4 KB
spindle speed	10000 RPM
head start time	1 ms
track to track seek time	0.002 ms

Question: how much time would be required, on average, to read a file of size 2.5 MB, assuming the file is stored in contiguous sectors on adjacent tracks (since one track won't hold the whole file in this case)?

Example: Contiguous File Read

We must compute some values from the given disk parameters:

rotations per second:	**166.67**
clusters per track:	**256**
capacity of one track:	**1 MB**
sectors per cluster:	**8**

So an 2.5 MB file would occupy two full tracks plus 128 clusters on an adjacent track.

For reading the first full track, assuming average values:

- seek time = 1 + (8096/3)*0.002 = 6.40 ms
- latency time = (1/2)*(60000/10000) = 3.00 ms
- transfer time = (2048/2048)*(60000/10000) = 6.00 ms

} total: 15.40 ms

Example: Contiguous File Read

For reading the second full track, assuming average values:

- seek time = 1 + 1*0.002 = 1.002 ms
- latency time = (1/2)*(60000/10000) = 3.00 ms
- transfer time = (2048/2048)*(60000/10000) = 6.00 ms

} total: 10.00 ms

For reading the relevant sectors from the second track, assuming average values:

- seek time = 1 + (1)*0.002 = 1.00 ms
- latency time = (1/2)*(60000/10000) = 3.00 ms
- transfer time = (1024/2048)*(60000/10000) = 3.00 ms

} total: 7.00 ms

So the total time to read the file into memory* would be about 32.40 ms or about 0.038 seconds.

*** Actually this is the time to read the file into the disk buffer memory.**

Example: Fragmented File Read

What if the file is scattered all over the disk?

Then it will take a lot longer…for each cluster:

- seek time = 1 + (8096/3)*0.002 = 6.40 ms
- latency time = (1/2)*(60000/10000) = 3.00 ms
- transfer time = (8/2048)*(60000/10000) = 0.02 ms

} total: 9.42 ms

Since the file occupies (part or all of) 640 clusters, the total read time for a completely fragmented copy of the file would be about 6028.9 ms or about 6 seconds.

So this would take almost 160 times as long as if the file were stored contiguously.

Sector Read vs Single Byte Read

The programs you have implemented simply perform read/write actions, usually on single variables which may store from 1 to a few hundred bytes of data.

Why do typical disk controllers perform read/write operations in sector-sized chunks? Performance… given the disk system described earlier:

time to read one full track: 15.40 ms

time to read one full cluster: 9.42 ms

6.40 ms + 3.00 ms + (1/256)*(60000/10000)

time to read one byte: 9.40 ms

6.40 ms + 3.00 ms + (1/512)*(1/2048)*(60000/10000)

The time to read one sector is only slightly more than the time to read a single byte, largely because the seek and latency times dominate the transfer time.

What's the moral??