Hard Disk Drives (HDDs)

HDD (software) interface

Modern conventional hard drives export a linear address space. In other words, they present the OS with the illusion of an array of *n* logical block addresses (LBAs), numbered from *0* to *n-1*, that the OS can access by block number. The OS must read/write at the granularity of blocks.

Most modern hard drives are "advanced format" HDDs, which means that their sector size exceeds the older standard of 512 bytes. A common physical block size is 4 KiB (4096 bytes), which matches the common OS page size.

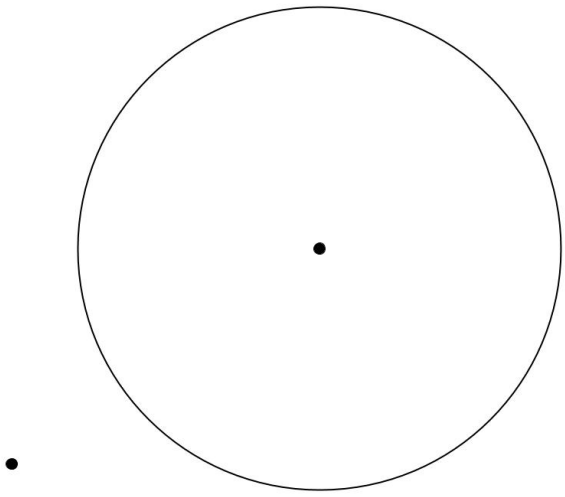
HDD Guarantees

- Sector writes are atomic: either the entire sector is written, or the previous contents remain unchanged.
  - When a multisector write is only partial completed, it is called a *torn write*, which is something that software at higher levels must be careful to handle.
- "**Unwritten Contract**": accessing two blocks with LBAs that are near to each other is faster than accessing two blocks with LBAs that are far from each other (i.e., sequential reads/writes are faster than random reads/writes)

Disk Parts and Geometry

Disks are essentially a collection of spinning metal surfaces, and as they spin, an arm moves from the center to the outer edge to read and write data.

1. Draw and label a 2D top-down view of a disk. Draw your disk with 1 **disk arm**, 3 **tracks**, and 8 **sectors** per track. (Place LBA 0 on the outermost track, and let the disk rotate counter-clockwise)



Breaking down an I/O

To read or write data from the platter, the disk arm must pass over the target sector. There are two parts to any I/O:

- Setup
- Transfer

We can think of the *setup* as all the work necessary to locate the disk arm at the appropriate physical location, and the transfer as all of the work that takes place starting from the moment that the transfer of the first byte begins.

1. The two components of the setup cost are the **seek** and the **rotational delay**. What physical actions correspond to each component?
2. On average, how many rotations does a given I/O's setup phase require?
3. Suppose that the disk spins at 7,200 RPM. How much time is a single rotation?
4. Give ballpark (order of magnitude) costs (latency, in ms) for a commodity HDD to complete

- one disk seek:
- one disk rotation:

Other Physical HDD Considerations

Geometry

Disk geometry affects the costs of I/Os in interesting ways.

1. What is meant by **track skew**, and what problem is it designed to solve?
2. Why are many conventional hard disks divided into **zones** (a zone is a contiguous collection of tracks; the tracks comprising a zone each have the same number of sectors, but each zone has a different number of sectors per track)?

Caching

HDDs have their own internal buffers (disk buffers are *O(10s of MiBs)*)

1. How might the disk's internal buffer be used to speed up certain reads (hint: locality)?
2. How might the disk's internal buffer be used to speed up writes?
3. Which of the above two optimizations (if any) are dangerous?
4. Explain the difference between write-back and write-through caching.

Scheduling

If the OS and disk were to attempt to immediately satisfy every I/O request, and to strictly complete requests in the order that they were received (i.e. FIFO request scheduling), performance would likely be poor. Why? Multiple applications compete with each other for use of the disk, and they must all share one disk head to read/write data. Further, no single application's requests will exhibit *perfect* locality (metadata is often separated from the data it describes, so even sequential data requests may require seeking: the file system must first read the metadata required to *find* the location of the target data). Given these challenges, we need more sophisticated strategies to order requests; we want to avoid as many unnecessary disk seeks as possible. A good scheduling algorithm should strive to:

- Maximize overall system throughput (the system as a whole should make the most possible progress in a given time window), and
- Maximize fairness (no one application should suffer from **starvation** while other applications make progress)

The textbook describes the principle of **shortest job first** (SJF). As the scheduler receives a series of requests, a queue may build. The queue can be (re)ordered in a variety of ways.

1. What is the relationship between *shortest-seek-time-first* (SSTF) and *nearest-block-first* (NBF)?
2. How does the "elevator" (or SCAN or F-SCAN or C-SCAN or ...) algorith related to both of the goals above (e.g., maximize system throughput and maximize system fairness)?

The anticipatory scheduler is alluded to in the text: an anticipatory scheduler will pause for a short time before dispatching an I/O from the queue. If a new I/O arrives, the scheduler has more information to help it optimize its I/O ordering. 3. Deliberately slowing down seems like a counter-intuitive strategy, but it works well for disks. Why might the anticipatory scheduler be a better strategy for a HDD than for a faster (lower latency / higher throughput) device?

More complex scheduling algorithms may prioritize certain applications over others. 4. What entity should be responsible for deciding an application's priority? What privileges should be necessary to declare your task as more important than others?

Hand-in Question

Modern conventional hard drives export a linear address space. In other words, they present the OS with the illusion of an array of *n* logical block addresses (LBAs), numbered from *0* to *n-1*, that the OS can access by block number.

OSTEP Chapter 37 briefly describes an HDD "unwritten contract": accessing two blocks with LBAs that are near to each other is faster than accessing two blocks with LBAs that are far from each other.

1. Assuming that disks are as described in the text, why is this (likely) the case?
2. Suppose you are given a hard disk drive. *Very briefly* describe how you might experimentally test this unwritten contract (not that you need to write code, but you may want to describe in prose what some code might do in terms of reading and writing LBAs)?