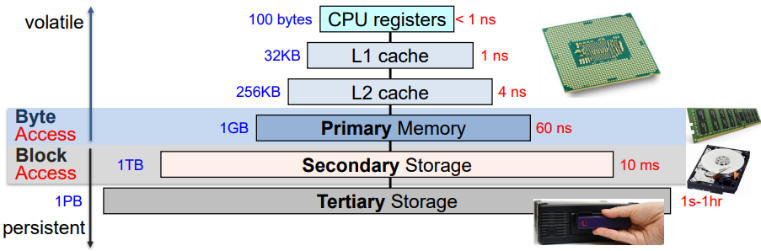
Operating Systems

Secondary Storage

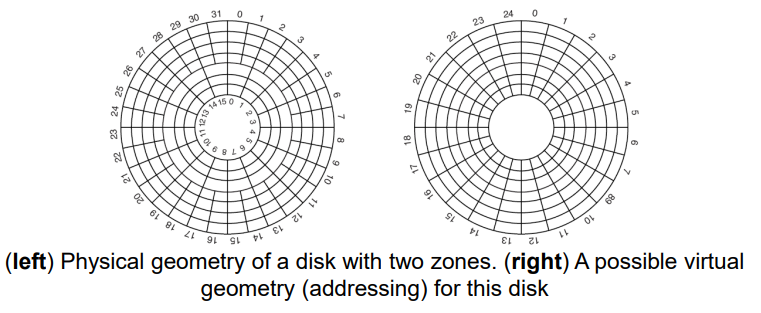
Traditional Secondary Storage

Data is typically accessed in blocks rather than bytes and the CPU cannot directly access the secondary storage.

Secondary storage is typically very large, very cheap and very slow, but it keeps its state even when unpowered making it vital for systems, It’s also very reliable, rarely failing.

Magnetic Diskss

Magnetic disks work by writing bits on magnetic platters in concentric circles called tracks. The most common types of storage using magnetic disks are floppy disks and HDDs.

The physical geometry and the addressing of magnetic disks can differ quite a lot. Physically the disk is split into concentric circles (tracks) and radial sectors (sectors), as the disks are spun and the radius of the circles changes but the angular velocity is the same, the density of bits can also change as you move to tracks further to the edges of disks. Virtually, we aren’t concerned with this and don’t consider the density at all, only worrying about the logical block address (LBA) of the data we want.

The performance of these disks depends on:

* Seek time – the time it takes for the read head to move to the right track
* Rotation (latency) – the time it takes for the desired sector to rotate under the read head
* Transfer time – the time it takes to transfer data from the surface of the disk to the controller

When using a disk the OS should try to minimise all of these costs (specifically seek and rotation).

Performance

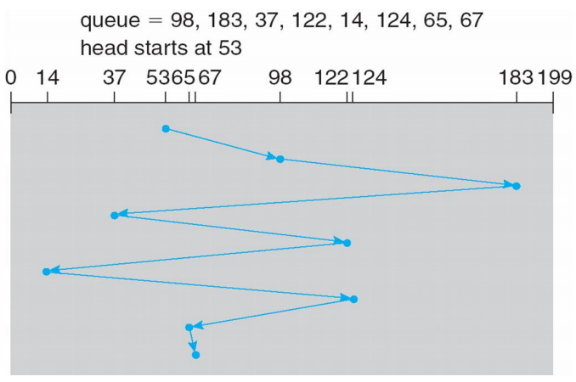
The OS may increase the file block sizes and co-locate ‘related’ items (put related data close together on the disk) to reduce seek time to reduce seek time.

The OS may also keep data or metadata cached in memory to reduce the need to read from the disk, however this takes up valuable memory. The OS could also fetch blocks before they’re needed (pre-empting) as a way to hide the slow disk.

Performance Via Block Scheduling

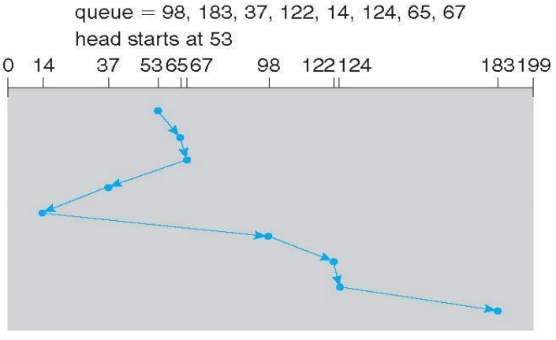
Applications request data access to the OS, which maintains request queues. The OS generates transfer commands to and from the disk(s) which imply seeks, rotation waits and data transfers but leaves the specifics to the device controller.

We can reduce the waiting time for the applicating by modifying the order in which the requests are satisfied (normally based on the location of the data), there are many block-scheduling algorithms:

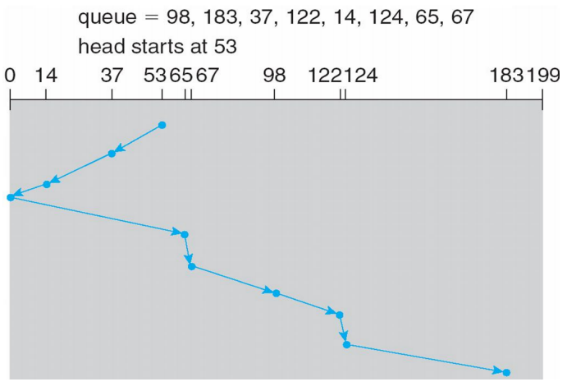
* FCFS – First come first served (no scheduling)
* SSTF – Shortest seek time first
* SCAN – elevator algorithm
* C-SCAN – typewriter algorithm

FCFS

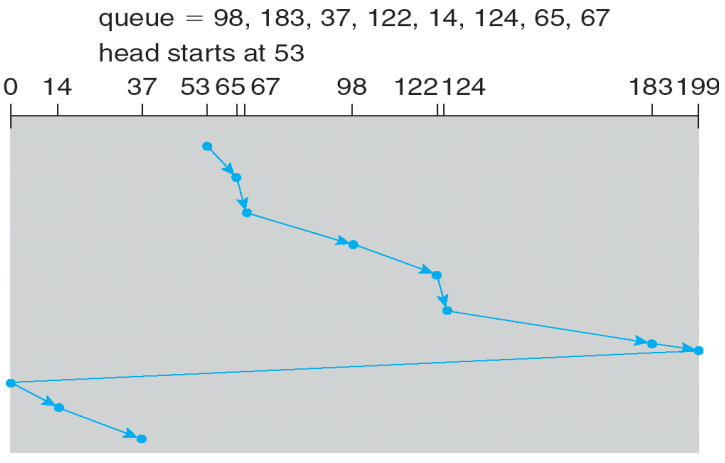
First come first served, the requests are serviced in the order they arrive. This is reasonable when the load is low but produces a long waiting time when the queue is long.

SSTF

Shortest seek time first, we service the location closets to the head next. This minimises arm movement (seek time) to maximise the request rate but unfairly favours middle (clustered) blocks.

SCAN

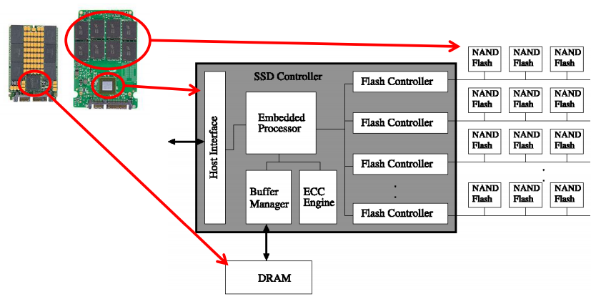
The disk arm starts at one end of the disk and moves towards the other services request as it goes until it gets to the other end. The direction is then reversed and servicing continues. IF request are uniformly dense then this leaves a higher density of requests at the opposite end to wherever the head is (as the head will have recently serviced all the ones at the current end).

C-SCAN

This works the same as scan except when the head reaches one end of the disk, it goes right back to the other, only ever servicing in one direction. This provides a more uniform wait time than SCAN and is known as the typewriter algorithm.

Selecting a Disk-Scheduling Algorithm

When there is one request, all the algorithms behave like FCFS. SCAN and C-SCAN perform better for systems with heavy load on the disk (causing less starvation). Performance depends on the number and types of requests which can be influenced by the file-allocation method and metadata layout.

The OS block-scheduling algorithm is made as a module of the OS to make replacing it easy, linux has many different algorithms.

Solid-State Drives

Solid state drives are a more modern, much faster, type of storage than magnetic disks. They work similarly to RAM just slower and permanent. There are many different technologies that can be used for solid-state drives and many different interfaces that they can use.

SSD Reads

The unit of a read for SSDs is a page, typically 4kB, the speed of these reads varies a lot, capable of roughly 100,000 reads a second with 10-100 micro-second latency (50-100 times better than disk) and 50-500MB/s read throughput (at least 1-10 times better than disk).

Another benefit of SSDs is that access time is (essentially) independent of the device’s geometry and as such a block scheduler is not needed, normally just using an FCFS queue.

SSD Writes

The unit of a write for SSDs is also a page but writing to an SSD is often slower than reading from one in every aspect. Additionally Flash media must be erased before it can be written to, the unit of which is a block (64-256 pages) which usually takes around one millisecond to do (depending on the technology), additionally memory can only be erased a certain number of times before it becomes unusable (10,000 – 1,000,000 times).

To extend the lifetime we require a flash translation layer (FTL) implemented in firmware to ensure the entire device wears out at the same rate (rather than a specific block wearing out quickly and rendering it useless when most of the drive is fine).

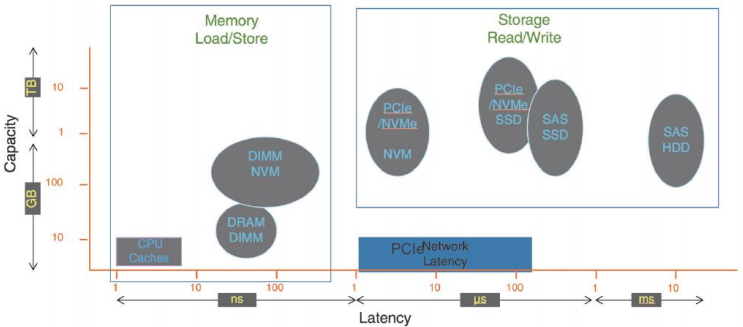
SSD Vs HDD

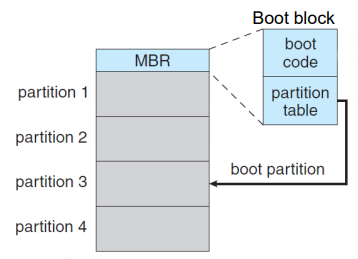
As of march 2020, flash SSDs cost at least £1/GB with 1TB drives costing around 100GB whereas a 1TB HDD costs around £35.

SSDs are typically more energy efficient using only 1-2 watts as opposed to roughly 10.

SDDs also have no moving parts making them more physically durable, HDDs cannot work correctly if subject to acceleration.

New And Old Secondary Storage Vs Primary Storage

Non-Volatile Memory (NVM) and Persistent Memory (PM, PMem) are non-volatile/persistent and have a main memory form factor (looking like main memory). They are also VERY fast. 

Storage Device Management

Just storing the data on the device isn’t enough, we also need metadata. Before storing the data, the device needs to be initialized, low-level formatting preparing it to hold data and metadata as well as partitioning.