【OS】Day40(2)

Class	Operating System: Three Easy Pieces
Date	@February 21, 2022

【Ch37】Hard Disk Drive(2)

37.4 I/O Time: Doing The Math

We can now represent I/O time as the sum of three major components:

$$T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$$

Now that the rate of I/O($R_{I/O}$), which is often more easily used for comparison between drives is easily computed from the time. Simply divide the size of the transfer by the time it took.

$$R_{I/O} = Size_{Transfer}/T_{I/O}$$

Assume there are two workloads we are interested in.

The first known as the random workload, issue small reads to random locations on the disk. Random workloads are common in many important applications, including database management systems.

The second, known as the sequential workload, simply reads a large number of sectors consecutively from the disk, without jumping around.

	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

Figure 37.5: **Disk Drive Specs: SCSI Versus SATA**

We take two drives, Cheetah 15K.5, a high-performance SCSI drive, and Barracuda, a drive built for capacity.

The two components of the disk drive market are “high performance” and “capacity”.

We can start to calculate how well the drives would do under our two workloads outlined above. Let’s start by looking at the random workload. Assuming each 4KB read occurs at a random location on disk, we can calculate how long each such read would take. On the Cheetah:

$$T_{seek} = 4\ ms,\ T_{rotation} = 2\ ms,\ T_{transfer} = 30\ microsecs \qquad (37.3)$$

The average rotational delay is calculated from the RPM directly. 15,000 RPM is equal to 250 RPS thus each rotation takes roughly 4ms. On average, the disk will encounter a half rotation and thus 2ms is the average time. The $R_{I/O}$ for Cheetah is about 0.66MB/s.

Finally, the transfer time is just the size of the transfer over the peak transfer rate.

Thus, the $T_{I/O}$ for Cheetah roughly equals 6ms.

The same calculation goes for Barracuda and it comes with a result of 13.2ms. and a $R_{I/O}$ of about 0.31MB/s.

Now let's look at the sequential workload. Here we can assume there is a single seek and rotation before a very long transfer. For simplicity, assume the size of the transfer is 100MB. Thus, $T_{I/O}$ for the Cheetah and Barracuda is about 800ms and 950ms. The rates of I/O are thus very near the peak transfer rates of 125MB/s and 105MB/s.

	Cheetah	Barracuda
$R_{I/O}$ Random	0.66 MB/s	0.31 MB/s
$R_{I/O}$ Sequential	125 MB/s	105 MB/s

Figure 37.6: **Disk Drive Performance: SCSI Versus SATA**

37.5 Disk Scheduling

Because of the high cost of I/O, the OS has historically played a role in deciding the order of I/Os issued to the disk.

More specifically, given a set of I/O requests, the disk scheduler examines the requests and decides which one to schedule next.

Unlike job scheduling, where the length of each job is usually unknown, with disk scheduling, we can make a good guess at how long a "job"(i.e. disk request) will take.

By estimating the seek and possible rotational delay of a request, the disk scheduler can know how long each request will take, and thus (greedily) pick the one that will take the least time to service first.

Thus, the disk scheduler will try to follow the principle of SJF(shortest job first) in its operation.

SSTF: Shortest Seek Time First

One early disk scheduling approach is known as shortest-seek-time-first(SSTF). SSTF orders the queue of I/O requests by track, picking requests on the nearest track to complete first.

For example, assuming the current position of the head is over the inner track, and we have requests for sectors 21(middle track) and 2(outer track), we would then issue the request to 21fist, wait for it to complete, and then issue the request to 2.

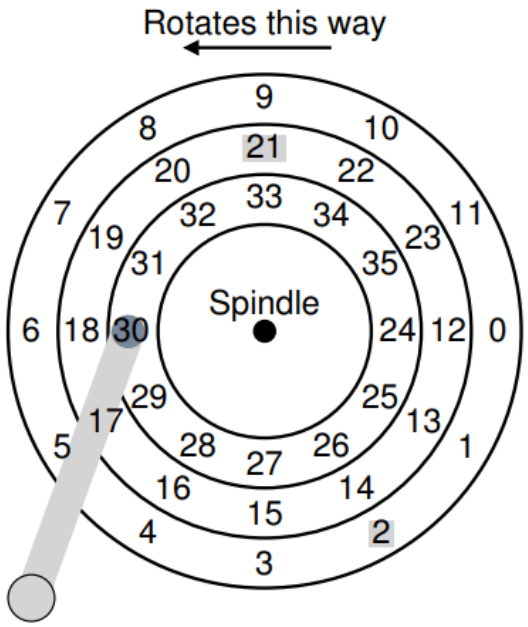

Figure 37.7: **SSTF: Scheduling Requests 21 And 2**

However, this approach can cause starvation. If there were a steady stream of requests to the inner track, where the head currently is positioned. Requests to any other tracks would then be ignored completely by a pure SSTF approach.

Elevator(aka SCAN)

The algorithm originally called SCAN, simply moves back and forth across the disk servicing requests in order across the tracks. Let's call a single pass across the disk a sweep.

Thus, if a request comes for a block on a track that has already been serviced on this sweep of the disk, it is not handled immediately, but rather queued until the next sweep.

C-SCAN is a common variant of SCAN algorithm, short for Circular SCAN. Instead of sweeping in both directions across the disk, the algorithm only sweeps from outer-to-inner, and then resets at the outer track to begin again. Doing so is a bit more fair to inner and outer tracks, as pure back-and-forth SCAN favours the middle tracks.

SPTF: Shortest Positioning Time First

The shortest positioning time first or SPTF scheduling is also called shortest access time first or SATF.

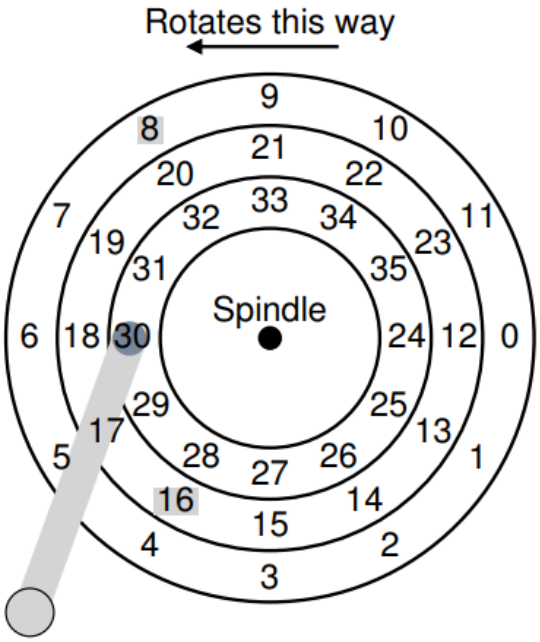

Figure 37.8: **SSTF: Sometimes Not Good Enough**

In the example, the head is currently positioned over sector 30 on the inner track. The scheduler thus has to decide: *should it schedule sector 16(one the middle track) or sector8(on the outer track)?*

It depends here. What it depends on here is the relative time of seeking as compared to rotation. If seek time is much higher than rotational delay, then SSTF are just fine. However, imagine if seek is quite a bit faster than rotation.

However, if seek is quite a bit faster than rotation, then in our example it would make more sense to seek further to service request 8 on the outer track than it would to perform the shortest seek to the middle track to service 16.