CS 390
Chapter 10 Homework Solutions

11.1 `Is disk scheduling, other than ...`

Yes. Even in a single-user environment, there may be multiple processes competing for access to the disk.

11.2 `Explain why SSTF scheduling tends ...`

Suppose that a disk has c cylinders. If you plot the average distance from a cylinder p and every other cylinder, you will see that the average distance is largest for cylinders near the edge of a platter, and smallest for cylinders near the middle of the platter, with the minimal distance occurring when $p = \frac{c}{2}$.

If cylinder requests are uniformly distributed over the surface of the disk[1] then the closest cylinder to a given cylinder is more likely to be in the middle of the disk.

11.3 `Why is rotational latency usually ...`

In an operating systems course, we only study the disk scheduling that can be done by the kernel. With current hard disk technology, the kernel has no way of discovering the rotational angle of the platters (i.e., the sector under the head.)

If the kernel could discover the current platter angle, it could calculate the actual amount of time required to service each request. This amount of time would be the sum of the seek time plus the additional amount of latency required for the correct sector to spin under the head. This calculation would be somewhat complicated by the fact the these two processes occur at the same time.[2]

11.9 `Give three reasons to use ...`

1. HDDs are cheaper than NVMs on a $/byte basis.
2. Blocks on an HDD have a potentially unlimited lifetime.
3. HDDs have faster access times than magnetic tape.

[1] Is this a reasonable assumption? Somebody should do a capstone on this.

[2] This would also be an interesting capstone: a disk scheduling algorithm that uses latency to optimize disk request times.

11.10 `Give three reasons to use ...`

1. NVMs have faster access times than HDDs.
2. Because they don't have motors, NVMs use less power than HDDs.
3. Sleeping NVM devices do not require a spin-up time before they can be used. HDDs do.

Additional Exercises

A None of the disk scheduling algorithms discussed can avoid starvation except FCFS.

1. Explain why this is true.
2. Describe how starvation can occur with SSTF, and describe a way to avoid it.

B Suppose a disk has 5000 cylinders, numbered 0 (innermost) to 4,999 (outermost). The head is currently on cylinder 2150. The queue of cylinder requests is

$2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681.$

What is the total distance, in cylinders, that the head moves to satisfy all the pending requests under each of the following scheduling algorithms?

1. FCFS
2. SSTF

C Give an advantage and a disadvantage of using an SSD as a caching tier along with an HDD, when compared to just using an HDD as a computer's main secondary storage device.

D A RAID 4 scheme consists of four disks that store data using block-level striping, and a fifth disk which stores a parity block for the other four. How many *physical* blocks are accessed in each of the following operations?

1. A read of one logical block.

2. A write of one byte of a logical block.
3. A write of an entire logical block.
4. A modification of two bytes that span two contiguous logical blocks.

E Traditionally, the kernel treats all block requests equally, and schedules them in an attempt to minimize the average wait time. Suppose that a kernel decided to prioritize block requests depending on the type of data that was being transferred. Arrange the following types of data in the order that a kernel should prioritize it.

1. The write of a block that contains part of a temporary file created by an application.
2. The read of a block that contains a faulted-for page.
3. The write of a block that contains a page selected as a victim by the page replacement algorithm.
4. The read of a block that contains the text segment of a new process.
5. The write of a block that contains the assembly language code generated by a compiler.
6. The reading of a block from a file used by the system's GUI.

Answers to Additional Exercises

A 1. In each of the disk scheduling algorithms except FCFS, it is possible for two requests to be serviced in an order that is different than their arrival order. Since the request queue is constantly changing as processes request disk IO, it is possible for an early request to have its service postponed indefinitely because a later request is serviced before it.

2. SSTF can cause starvation if requests for cylinders that are close the RW head continuously arrive while there is an outstanding request for a cylinder that is far from the head. We could avoid starvation by aging each request.

   That is, the algorithm would take into account not only the distance of the requested cylinder from the current head

position, but also the amount of time the request has been in the queue. Older requests would be considered to have a smaller distance from the head than newer requests, regardless of the actual distance.

B

FCFS	13,011
SSTF	7,586

C If the kernel has a good caching algorithm, frequently used files (including applications, swapped out pages, and application data) will often be found on the SSD. Since secondary storage requests will be serviced more quickly, processes will spend less time on the wait queue, leading to a more responsive system.

However, if the system runs mostly CPU-bound jobs, most processes will be found on the ready queue rather than on the wait queue. A fast secondary-storage system doesn't help processes that spend most of their time on the ready queue. Adding an SSD will add cost to the system without adding a benefit.

D In this scheme, a logical block is mapped to four physical blocks. Keep in mind that each change to one or more of the physical blocks that make up a logical block requires the RAID hardware to update the parity block.

1. Four physical blocks are accesses
2. Remember that the kernel deals in logical blocks. It typcial has no knowledge of the physical disks that the RAID device is built from. This operation requires nine physical block accesses. Four to read the physical blocks that make up the logical block. Four more to write the physical blocks back to disk. Finally, the parity block must be updated by an additional write.
3. Five physical block accesses, all writes. We don't have to read the logical block first, as we did in 2 because we are completely replacing the contents of the logical block.
4. Eighteen physical block accesses: eight reads to get the two logical blocks, and eight writes to write them back to disk after the logical blocks have been modified, and two to update the parity blocks.

E This is just a thought experiment, so there is no one correct answer. Here are some things to consider.

1. You might want to prioritize the disk operations involved in supporting demand paging, so you don't completely destroy the system's effective access time when the total demand for memory exceeds physical memory.
2. If we want a system to be responsive, new programs have to launch quickly. We might want to give high priority to secondary storage operations that are involved in process creation. This consideration might also interact with item 1 if the system uses prepaging during process creation.
3. Similarly to item 2, we might give high priority to files used by the system's UI.
4. Compilers typically write the assembly language version of the program to a file, but that file is quickly translated to machine code and then deleted. If you postpone writing that file to disk, it may be deleted quickly and the system can avoid writing it at all.