Contents

About these reading notes

These are my own personal reading notes that I took (me, Franklin) as I read the textbook. I'm providing these to you as an additional resource for you to use while you're reading chapters from the textbook. These notes do not stand alone on their own — you might be able to get the *idea* of a chapter while reading these, but you're definitely not going to **get** the chapter by reading these alone.

These notes are inconsistently all of the following:

- Me summarizing parts of the text.
- Me commenting on parts of the text.
- Me asking questions to myself about the text.
    - … and *sometimes* answering those questions.

The way that I would expect you to read or use these notes is to effectively permit me to be your inner monologue while you're reading the textbook. As you're reading chapters and sections within chapters, you can take a look at what I've written here to get an idea of how I'm thinking about this content.

Chapter 37: Hard disk drives

- Now we're talking about I/O, cool. You can *consider* looking at chapter 36, it describes a generalization of input and output devices where this chapter focuses specifically on hard drives as I/O devices, and the related I/O scheduling topics.
- This chapter focuses heavily on spinning platter hard drives ("spinning rust"). By the time you're at the end of this chapter, try to decide whether or not a lot of the issues that you looked at here apply to solid state drives (SSDs), where there *are* no platters, and access times are effectively uniform.

37.1 The interface

- Sectors. If you're also in COMP 3370 right now, this is **another** word to describe "chunks" of memory (on top of frames, slots, blocks, words, pages).
- We're eventually going to get to file systems (how software organizes data physically on a disk), but try to think about this now: how might software that organizes data physically on a disk use this atomicity guarantee that a 512-byte sector is either fully written or not written at all?
- The authors make a statement that sequential reads or writes will be faster than random reads or writes. Can you convince yourself that this is true based on figure 37.1?

37.2 Basic geometry

- This section lists a lot of facts about circles and stuff. Literally geometry. How does π fit into this picture?
  - Seriously, can you, for example, *calculate* the time that a single rotation takes of a disk platter given what the authors have provided here? What additional information do you need if you can't calculate it?

37.3 A simple disk drive

- As you are reading through this section, *try* to think about how software that organizes data physically on a disk (a file system) can use the information that the authors are describing to inform itself about how and where to physically put bytes on the disk.

Single-track latency: The rotational delay

- Convince yourself that
$$\frac{R}{2}$$
is the average delay time here.
- If accessing sector 5 is the worst case, what's the best case?

Multiple tracks: seek time

- Acceleration? Coasting? Deceleration? What is this, physics? The Fast and the Furious? 2Fast4Disks.
- The important takeaway from this should be that each of these different "phases" take a certain amount of time, and that amount of time very quickly adds up to milliseconds of access time. Think about this for a second: I/O is a "blocking" operation, the OS and the scheduler will do things like stop scheduling jobs that are waiting for I/O to finish. Milliseconds doesn't seem like a very long time. How much work can a process do in milliseconds?

Some other details

- We talk about caching extensively in COMP 3370, the concepts are all effectively the same here (if you've taken or are taking COMP 3370).

  - COMP 3370 question: write back vs write through in terms of cache → memory really only considered performance, but now write back vs write through has *real* side effects, like something happening during a power loss. In terms of a between CPU and main memory cache *do* we need to consider things like power loss?

37.4 I/O time: Doing the math

- As you're starting to look at these measurements, try to think about how this is related to what we thought about in terms of scheduling (e.g., with scheduling we considered turnaround time, arrival, start time, finish time, etc).
  - Remember how we had those unrealistic simplifications for scheduling? One of them was that we knew how long a job would take to complete. In terms of I/O: **do** we know how long a job will take to complete?
- A 1TB drive is a drive "built for capacity". LOL.
- Based on the numbers and formulas that you see here, what has a bigger effect: average seek time or RPM?
- The RPM change (and seek time change) from the Cheetah to the Barracuda are both *approximately* 2x, but the sequential rate of transfer reported on pg 8 is nowhere near 2x. What's the deal with that?
- On pg 9: Oh no. It's calculus. … or wait, is this physics? Oh no.

37.5 Disk scheduling

- What is this, processes?
- Think about this for a second: Logically, it would seem to make sense that I/O be serviced on a first-come-first-serve basis (e.g., process 1 makes a request before process 2, therefore process 1's I/O job should be completed before process 2's I/O job). But *does it*? Think back to the geometry of a disk. Think back to the unrealistic limitations that we put onto *processes*. Are the limitations that we put on processes still unrealistic when we think about them in terms of servicing I/O jobs?
- The authors don't state this explicitly, but where do you think the disk scheduler lives? In the OS or in hardware? You *might* want to look back at chapter 36 briefly to see how an OS asks an I/O device for information, specifically section 36.3 on pg 4 in chapter 36.

SSTF: Shortest seek time first

- Which process scheduling policy does this most closely resemble? Does this I/O scheduling policy suffer any of the same problems that the process scheduling policy suffered? Consider this for both SSTF and NBF.
- Starvation!

Elevator (aka SCAN or C-SCAN)

- The section is titled "Elevator" where SCAN and C-SCAN are "aka", but only later do the authors actually use the word "elevator".
  - Really convince yourself about what's happening here. Like literally visualize an elevator going up and down in a building.
- *Does* this resemble any of the scheduling policies that we've seen?
- How exactly does this solve the issue of starvation and fairness? Think back to how starvation and fairness were addressed in terms of scheduling algorithms, and think back to how they were addressed in terms of locks. Is the same solution used here (even transitively or implicitly)?

SPTF: Shortest positioning time first

- Which process scheduling policy does this most closely resemble? Does this I/O scheduling policy suffer any of the same problems that the process scheduling policy suffered?
- In terms of thinking about this like a process scheduling policy, what exactly would "positioning time" or "access time" be in terms of a process? Is there a relationship here?

Other scheduling issues

- Oh wait, the authors *do* talk about where disk scheduling happens. Neat.
  - Do you think that a hard drive has an OS on it?
- This is related to the topic, but beyond the scope of our course: can you convince yourself that *waiting* for more I/O is a better choice than immediately issuing I/O requests when you receive them?