NTNU
Kunnskap for en bedre verden

Department of Computer Science

TDT4186 - Operating Systems

Exercise 6

I/O Scheduling, DMA & RAID

Professor:
Di Liu
Teaching Assistant:
Roman K. Brunner
Student Assistants:
Eik Hvattum Røgeberg
Halvor Linder Henriksen
Ole Ludvig Hozman
Daniel Hansen

Handout: 20.03.2023

Introduction

So far, we only worked on volatile parts of the operating system, meaning that once power is gone, the state is also gone. While this is fine for some small systems, for many systems, we want to save state persistently, e.g. photos you have taken. For that we use disks but might also use other devices such as printers and USB sticks. Communicating with those devices is called input/output, as information flows in and out of the system. The I/O subsystem handles this, and many decisions depend on the type of device and the desired properties. This exercise will focus on hard disks, memory mapping, and direct memory access I/O.

1 I/O Scheduling

For the following exercises, assume the disk organisation as in Figure 37.7, Page 10[1].

1. Assume that the arm and head are in the position as shown in figure 37.7 to begin with. The entries to be accessed are 31, 32,33, 7, 8, 12, and 23. Assume that it takes the same amount of time to move the arm by one track, as it takes for one track to move to the next entry (e.g. if we read 30, the next entry that we can read is 20 on the middle track, as we need one time unit to read out and another time unit to move the arm, thus 19 is in the read position while the arm is moving).

   (a) Given the list of I/O operations, in which order would they be accessed given the Shortest Seek Time First (SSTF) approach? How many timeunits does it take to complete all I/O operations?

   (b) Would any other of the presented I/O scheduling algorithms behave differently?

2. Given an average seek time of 4ms, what would be the worst-case seek time you would expect to see?

2 RAID

Even though many systems nowadays run on SSD's, magnetic drives are still in high demand due to the fact that their storage is very cheap and we don't always need the raw speeds of SSD's. While RAID configurations used to be about making slow magnetic HDDs a bit faster and more reliable, the focus is now mainly on the reliability part of the equation.

Assume the numbers given in Figure 37.5 in chapter 37.4[2] for the following exercises.

1. For the first subtask we are still looking at how much more throughput we might be able to get out of a raid configuration. Assume a RAID-0 configuration, how many additional disks do we need when using the Barracuda HDD compared to the Cheetah in order to match the throughput? Assume two RAID-0 systems one with only Cheetah's, the other with only Barracudas. We are considering the steady-state sequential throughput. How does the amount of storage compare between the two systems?

2. Assume that our system runs RAID-0 on 20 disks. We assume an annualized failure rate (AFR) of 1% for every single disk, which means there is a 1% probability that any given disk fails during the period of one year. How likely is it that we will lose data over a period of three years?

3. We want to improve the reliability of our system by splitting up our 20-disk system into 4 RAID-4 systems with 5 disks each. How much memory do we have overall and what is the reliability of such a configuration compared to the single 20-disk RAID-0 system in the question above? Assume we don't replace broken hardware.

[1] https://pages.cs.wisc.edu/~remzi/OSTEP/file-disks.pdf
[2] https://pages.cs.wisc.edu/~remzi/OSTEP/file-disks.pdf, page 7

3 I/O Interaction

You are given a new piece of I/O hardware that enables the user to input and output odours. It has a very simple interface, with one 64-bit register each for Status, Command and Data. The Status can either be `UNKNOWN`, `BUSY`, `READY`, and `MODE`. If `MODE` is set to 1 it is reading odours, if it is set to 0 it is set to outputting.

1. Given the description above and the three commands `INIT`, `WRITE` and `READ`, what would a simple programmed I/O driver for the device look like? Assume that `INIT` initializes the device to whatever flags have been written to the DATA registers.

2. After releasing the device with your simple driver, you receive user complaints that when watching a movie with odour enabled, the machine's performance greatly degrades (The machine gets very hot, stuttering images and low responsiveness of the system overall). What could cause that problem and how could you improve your driver? What changes would you need in the interface of the device?

3. For DMA engines, we need to allocate a buffer that the engine can either read from or write to. This means that we need to provide a way of how the DMA engine (and the program) can detect if it is reading valid data from the buffer. Imagine our device is writing new data to the buffer, how can the CPU know, which parts of the buffer are already ok to be read? Propose a solution to that problem.