Project 2 Write-up

Instructor: D. Kevin McGrath
CS 444, Spring 2017, Oregon State University

May 8, 2017

1 Design Plan

The scheduler has to have requests in order (i.e. a queue) in order for the dispatch to even run as displayed in the noop I/O scheduler. Thus, all of my pending requests are added to a queue based on cases. Adding is not an issue if the queue is empty, and if it isn't then we just check to see if its sector value is between it's front and back sectors before inserting the request. Otherwise, the request will just be added to the back of the queue. The dispatcher turned out to be relatively easy because all you have to do is dispatch whatever has the smallest gap (so the SSTF I/O scheduler's purpose), but since it's sorted it's always the next one that gets dispatched.

2 What do you think the main point of this assignment is?

The main point of this assignment was to apply what we learned about the scheduler and practice our parallel thinking. This project achieved that goal by providing a project where we had to juggle a system where we had limited resources for our threads thus we had to find a good way to share so that the job could be (somewhat) efficiently achieved. This is similar to the job of a scheduler and thus it served a dual purpose of giving some insight onto the mess the schedulers solve.In addition to that we had the kernel project where we had to implement a new scheduler. This had the sole goal of making us work with the kernel so as to help us better understand how schedulers operate in the context of a kernel.

3 How did you personally approach the problem? Design decisions, algorithm, etc.

Much of what was accomplished with our solution was done looking at noop's I/O and Deadline I/O schedulers as examples. With those examples, it only required a bit of effort and debugging to make it work. Overall, we pretty much do what noop does, exept in sectors are compared with eachother before being dispatched using noop's dispatcher as a reference in our solution.

4 How did you ensure your solution was correct?

I used the tried and true method of spamming print statements to ensure everything was working properly at each state of the solution. While a bit tedious, I think this was a decent method since the code jumps around so much, and the print statements let me track everything without needing to do much beyond putting in the statements.

5 What did you learn?

Dealing with kernel panics was a lession in patience, since one small error in the code can cause a wall of error messages.

6 Work Log/Version Control

all code work done on this kernel solution was done by Charles Henninger on May 8th, while Sibi worked on concurrency 2. Vin handled parts of the write up, and helped with code.