Shambhavi Srivastava

Lab 4- Linux Schedulers

Operating Systems

Table of Contents:

Abstract

The objective of this assignment was to spend some time investigating the behavior of Linux Schedulers, investigate three diverse Process Schedulers that are executed in the Linux OS. Basically, how they perform on three unique sorts of processes. These schedulers were evaluated with I/O processes, CPU-Bound processes and the combination of the two. The following will be achieved through benchmarks. We will have to design a series of benchmarks to evaluate the behavior of several scheduling policies in the Linux Scheduler.

The Process Schedulers that were assessed were First-In-First-Out Scheduler (FIFO), and the Completely-Fair Scheduler (CFS), and Round-Robin Scheduler (RRS). As we have read so far, all the three schedulers are pretty similar to each other, as they maximize overall CPU utilization. However, if we were to talk about the fastest, most efficient running scheduler for both the processes, it would have to be CFS, making it to be the best scheduler for CPU- bound processes, whereas FIFO and RR schedulers are the the best schedulers for the I/O processes.

Introduction

A scheduler is a must for the operating system to constantly keep on running. In this program, we are testing different conditions for each process that is running. Schedulers are used to a) minimize the average or peak turnaround time, exactly how long it takes a process to execute from the first entry to its last exit, also known as maximize fairness, meet deadline and ensure the priorities are adhered to. The three main schedulers assessed in this assignment, and as mentioned above in the abstract are- a) First In First Out, b) Completely Fair Scheduling c) Round Robin Scheduling. As mentioned, the benchmarking test will be examining these schedulers.

Method

To investigate the schedulers mentioned above, we ended up creating three different kind of programs for each process. For our CPU bound process, I went with using the pi scheduler to run the benchmark test. For our I/O bound process, I went with using the input file that includes reading and writing and an output file that writes the data. For our mixed process, I went with using the combination of the pi scheduler, and the input and output file. As mentioned above, we are using RR , FIFO, and Other. As a user, we need to decide what kind of processes needs to get

passed in order to test the parameters, and exactly what is the right scheduler for our programs to get tested with. As mentioned in the abstract, I have set my program to three different variables, i.e. a, b and c. A representing the lowest, b slightly higher(medium) and c, the highest. A representing 5 processes, b representing medium for 20 and c representing a high for 100. We later come down to the part of creating a test script.sh file, obviously using the bash scripting. We do this to test the programs for all mixed of different kind of scheduler, various numbers of process and its type. As declared, there were 10,000 iterations for every time the process ran. The data collected is represented through graphs down below. To talk specifically about scheduling, we have been talking about process of scheduling that is being used in this assignment, and to assess the different kind of schedulers, we wrote a program called schedule.c. Starting off, we set the process to the max priority for the given scheduler - the CFS scheduler, and then we set the new scheduler policy eventually the mixed processes. Next, it performs forking in which the given processes are getting forked off. Once it gets passed to the parent id of the child, for the command to execute, forking succeeds. If pid ---- 0, it would start calculating pi using statistical method across all of its iterations.

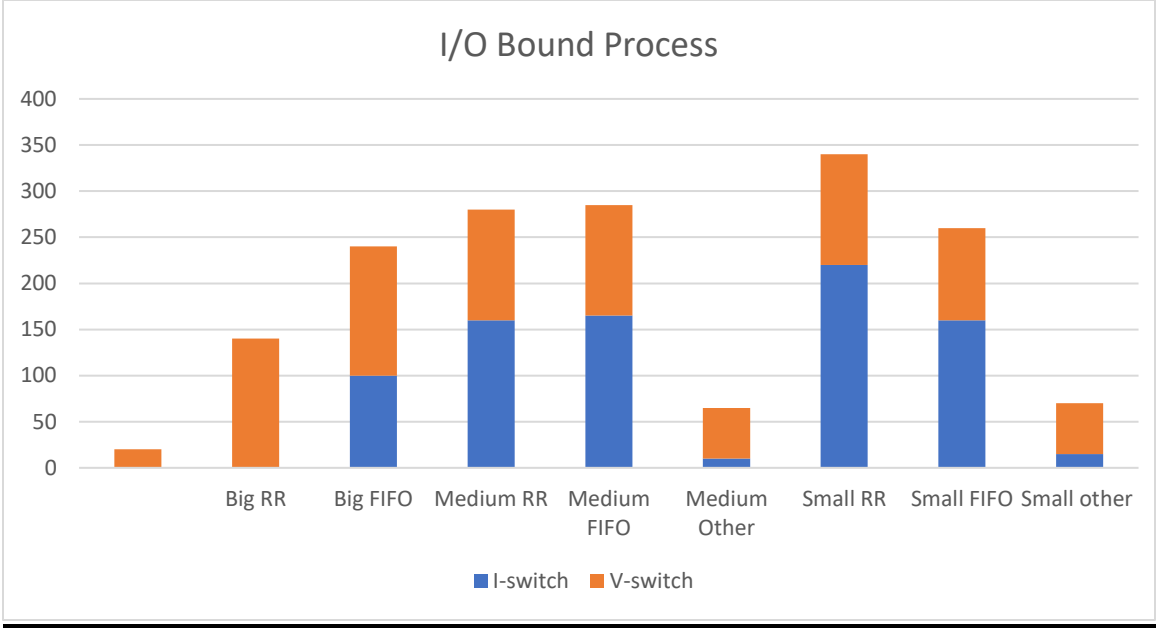
Analysis

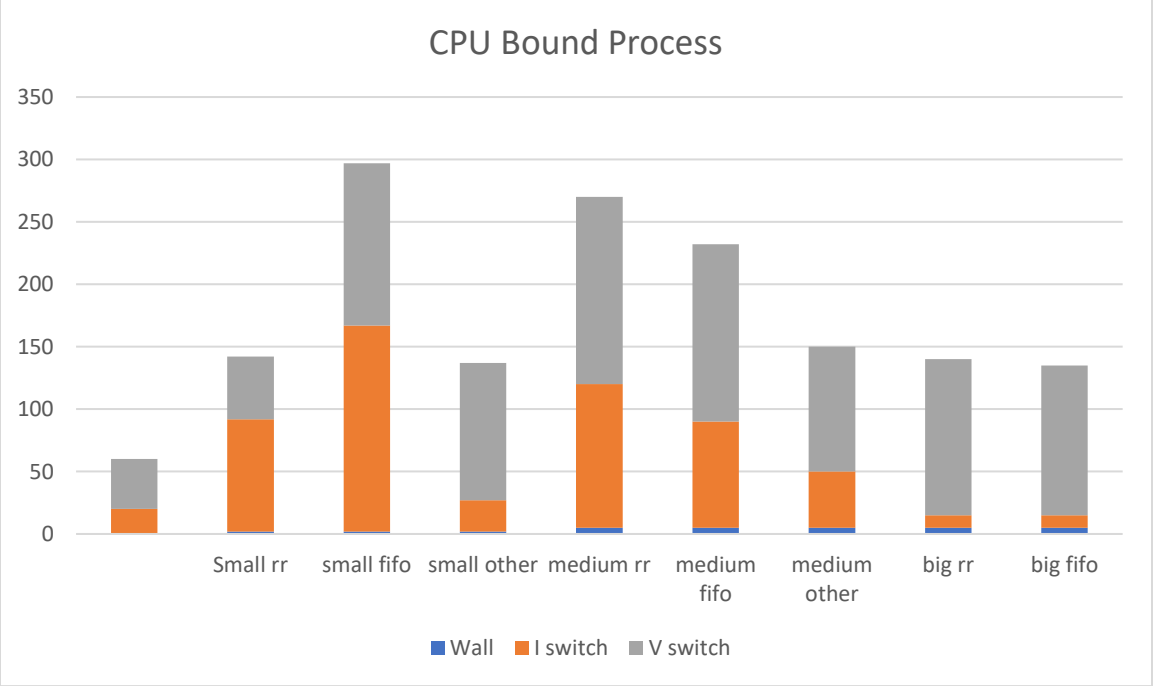
Right off the bat, I was excepting CFS to at the highest expense, going by how efficient it is and expected a good amount of context switching to occur for big running processes and eventually that was true. Talking about the I/O bound, I was not particularly shocked to see that a lot of interruptions were occurring, because we know that in comparison to CPU bound, I/O bounds are much slower.

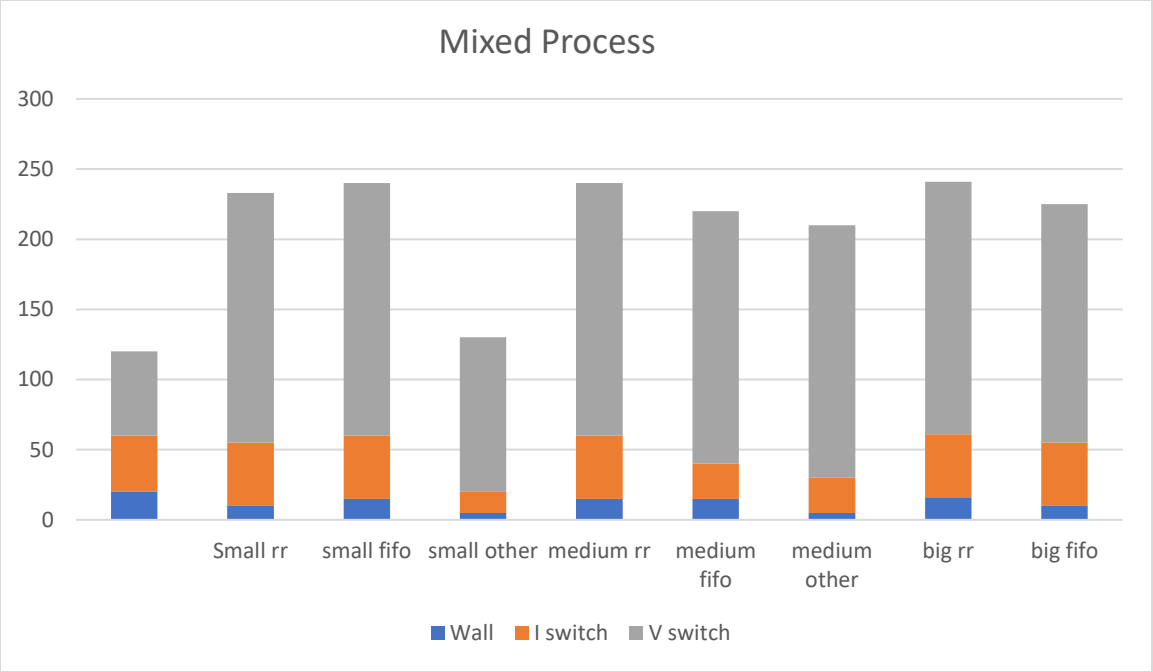
Conclusion

As predicted, it makes sense to say why CFS scheduler is the most efficient one to use when we talk about Linux Schedulers. We know that CFS schedulers simply sums the run times given each task and chooses the one to schedule with minimum sum, which is equivalent to choosing the task owed the most time on an ideal fair CPU according to derivation, and that's how it achieves fairness. However, the main thing to take away is each scheduler is getting a fair amount of time to run on the CPU, and by chance it takes too much time, another process will start right away as the scheduler will be hindering the process.

Results


I/O Bound Process
400
350
300
250
200
150
100
50
0
Big RR
Big FIFO
Medium RR
Medium FIFO
Medium Other
Small RR
Small FIFO
Small other
I-switch
V-switch


CPU Bound Process
350
300
250
200
150
100
50
0
Small rr
small fifo
small other
medium rr
medium fifo
medium other
big rr
big fifo
Wall
I switch
V switch


Mixed Process
300
250
200
150
100
50
0
Small rr
small fifo
small other
medium rr
medium fifo
medium other
big rr
big fifo
Wall
I switch
V switch