California State Polytechnic University, Pomona

Operating Systems Scheduling Algorithms

David Shin
CS 431 Operating Systems
Professor Atanasio
Due: 2/11/2018

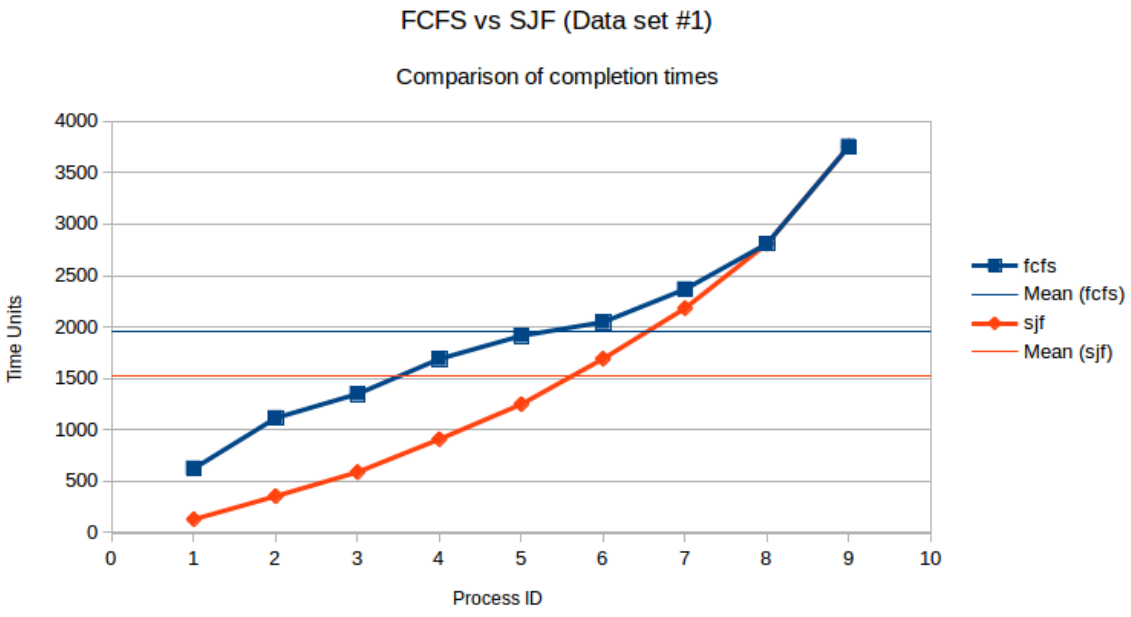
Introduction

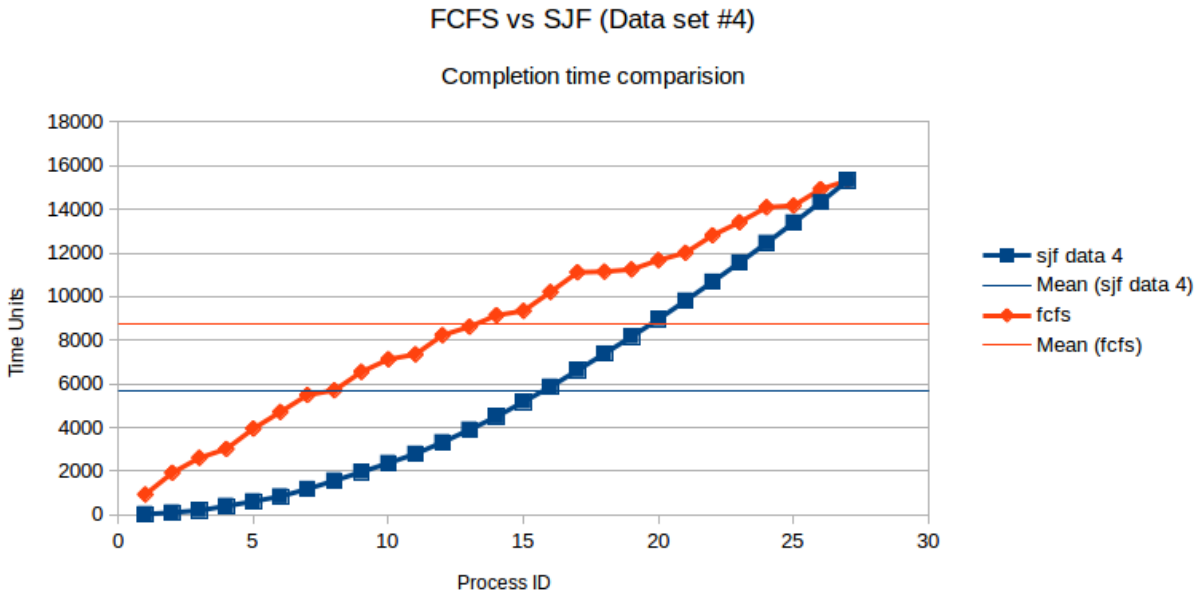
In this project, we implement 4 different types of scheduling algorithms to simulate how the operating system handles the execution of multiple processes. We will then analyze the performance of each algorithm with respect to test input data. The algorithms that are implemented are: First Come First Serve, Shortest Job First, Round Robbin, and Lottery. All algorithm, with the exception of FCFS and SJF will be using a time quanta, or the time allotted for a process to execute per cycle, of 25 or 50 time units. Swap time, or the time it takes to preempt one process and start another process is a constant time of 3 time units. We will look into how each algorithm performs with these restrictions in use.

Analysis

First Come First Serve vs. Shortest Job First

According to our test results from running both algorithms on data set #1, we can see that FCFS has an average completion time of 1963.11 time units and SJF has an average completion time of 1518.67 time units. On average, we can conclude that SJF has a faster execution time than FCFS. Presented below, our graph with more clearly depict our results. As data sets grow in size, we can see that the mean completion time is still faster for the SJF algorithm. The average completion time for FCFS in data set #4 is 8770.52 and 5685.41 in SJF.



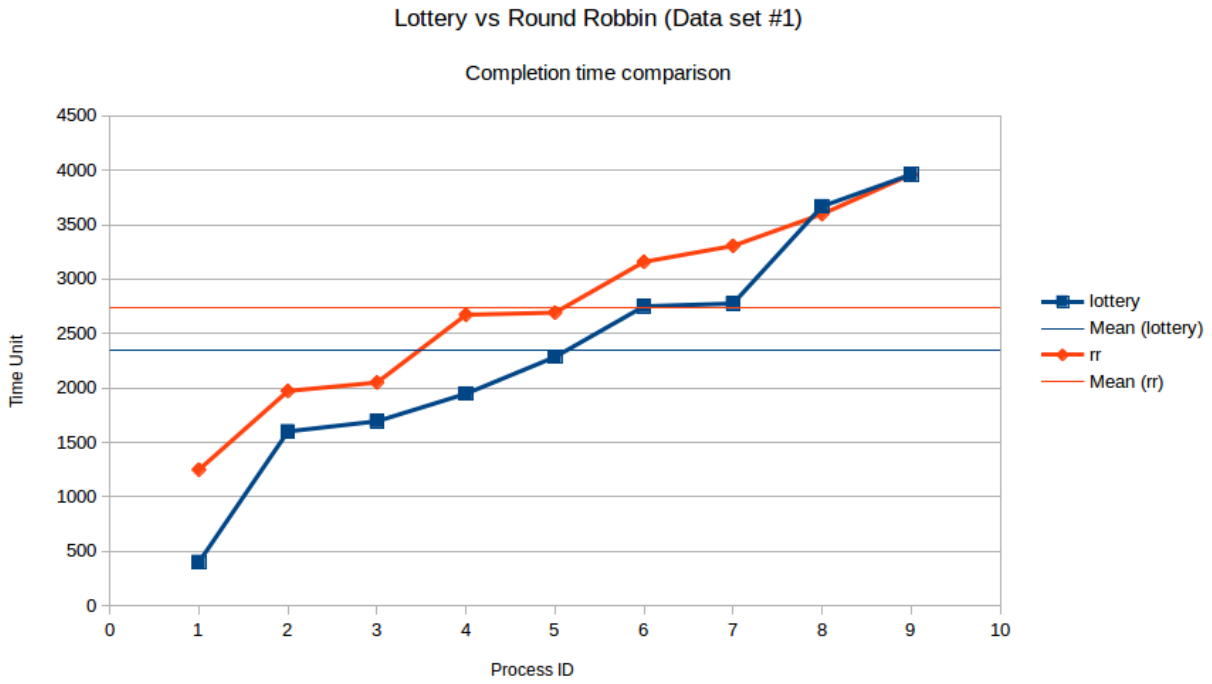


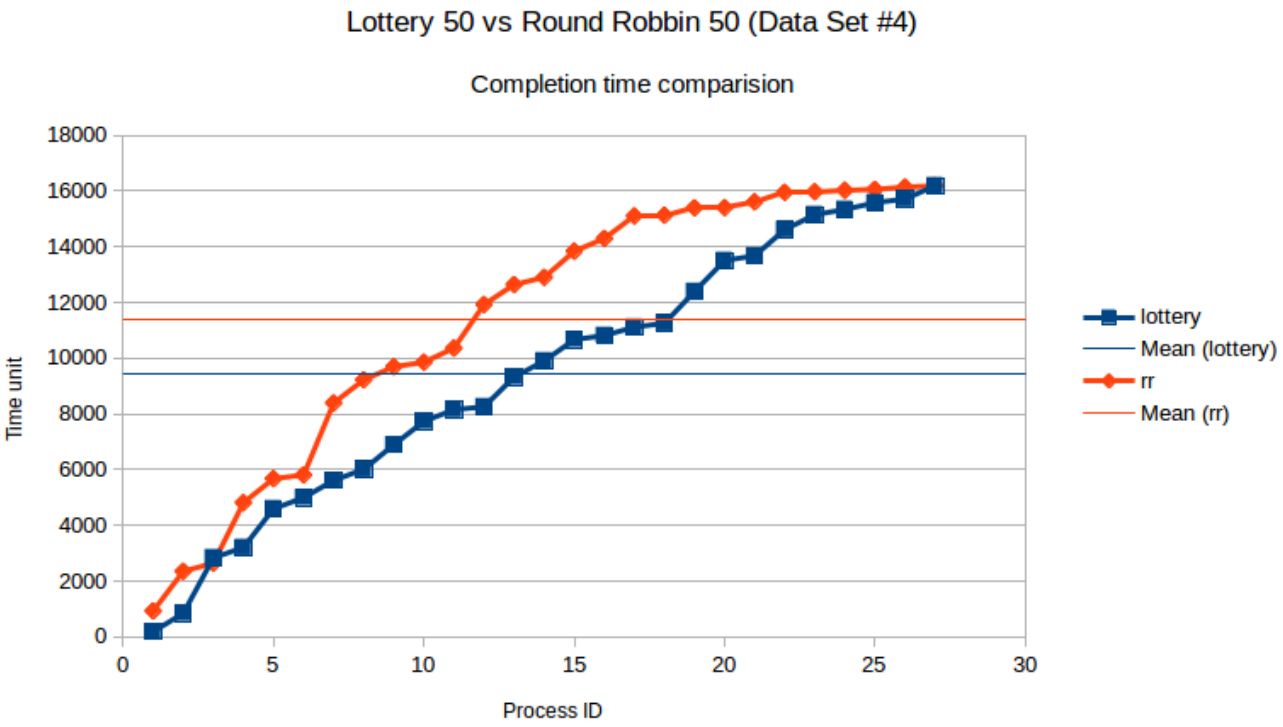
Round Robin 25 vs. Round Robbin 50

Next, we will analyze RR with a time quanta of 25 time units and RR with 50. Using test data #1, RR25 had an average completion time of 2917.44 time unit, while RR50 had an average completion time of 2740.11. Based on these results, it is enough to hypothesize that RR50 has a better average completion time than RR50. To test this hypothesis, we can compare both versions when running large data sets, for example data set #4. Using data set #4, RR25 has an average completion time of 12054.8 while RR50 had an average completion time of 11419.7. We can conclude that RR50 has an overall better completion time than RR25.

Lottery 50 vs. Round Robbin 50

Lastly, we will analyze the performance of the Lottery algorithm and Round Robbin with a time quanta of 50 time units. Using test data #1, Lottery had an average completion time of 2342.78 time units while RR50 had an average completion time of 2740.11 time units. Next, we will use a larger data set to verify if Lottery will perform better on average. Using data set #4, Lottery had an average completion time of 9428.96 time units, while RR50 had an overall average completion time of 11419.7. We can conclude that lottery, with a time quanta of 50 and given that the data sets are the same, will always outperform round robbin.




Lottery 50 vs Round Robbin 50 (Data Set #4)
Completion time comparision
Time unit
Process ID
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
0
5
10
15
20
25
30
lottery
Mean (lottery)
rr
Mean (rr)

Conclusion

We have analyzed the performance of 4 different types of scheduling algorithms and we can conclude that SJF will almost always have the fastest average completion time. However, SJF has a limiting drawback to it, which is the fact that a process with a large burst time must execute only after all processes with a shorter burst time have complete before it. This is a drawback because the process of the larger burst time may have greater importance than the processes with shorter burst time. Comparing SJF with FCFS, SJF will always have a faster average completion time. A solution to distributing execution time to processes in a more equal fashion would be to introduce round robbin or lottery. These such algorithms have the advantage that at least every process will have an equal amount of time to execute, such is called the time quanta. Comparing RR25 with RR50, RR50 has an overall better completion time than RR25. Comparing RR50 with lottery, lottery had an overall better completion time.

Links

https://www.github.com/davidws22/CS431-Project1