CSC3002F
OS2 Assignment

Christopher Blignaut BLGCHR003

Introduction

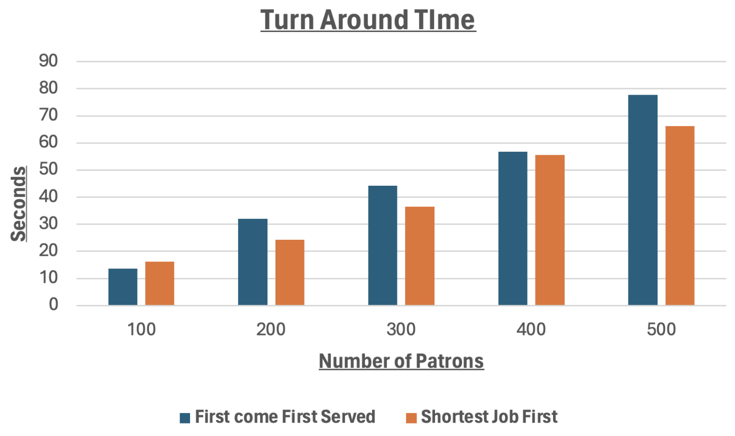
The aim of this report is to compare the Shortest Job First (SJF) algorithm and the First Come First Served (FCFS) algorithm to determine which algorithm is better for a CPU scheduler to implement. To do this, a scenario was created where the CPU was a barman and processes were patrons entering the bar. Each patron submits a drink order full of drinks (which represent jobs of the process) that the barman has to complete.

The algorithms' effectiveness are measured using the turnaround time, response time, waiting time and throughput metrics. This report aims to compare and contrast the metrics of said algorithms to determine the algorithm that has the better performance through not only the metrics mentioned above, but also through the determining of predictability and level of starvation that each algorithm is subject to.

Turnaround Time

Turnaround time is measured as the difference in time between the submission of a process to the completion of said process. In this program, this is represented by the difference in time between a patron submitting their first drink order to receiving their final drink. To maximise performance, we want to minimise this value as it means that our barman is completing orders in a smaller amount of time.

The average turnaround time of the patrons can be seen below:



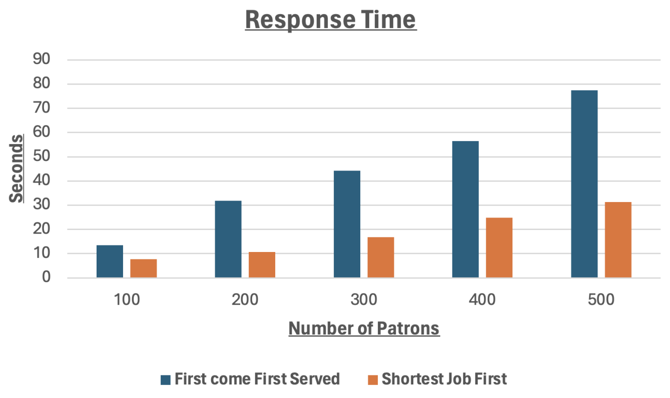
FCFS Variance: 859150142.3 ms^2

SJF Variance: 1648692237 ms^2

As can be seen, the FCFS algorithm has better performance for the smaller dataset of 100 patrons. However, as the number of patrons increases, the Shortest Job First algorithm performs better. There is a discrepancy at 400 patrons where the FCFS and SJF algorithms have very similar performance. This can be attributed to the random generation of drinks. The variance is quite high, however this is to be expected when measuring in milliseconds. This is to be expected as some patrons have large orders, while others have smaller orders. SJF is able to order the orders which in turn ensures that the time to fulfil the order is more predictable. Therefore, it is clear that SJF is more predictable with its turnaround time than FCFS as it has a lower variance. Therefore SJF results in the best overall performance for the turnaround time metric.

Response Time

Response time is defined as the time it takes for the CPU to produce a response to a request from the time said request was submitted. This is different from turnaround time as only part of the process needs to be completed. Despite their differences, to maximise performance, we want to minimise this value as well as it means that processes are attended to and do not suffer from starvation. This was calculated in the program as the time it took from the patron submitting their drink order to the time that the barman served them their first drink.



FCFS Variance: 858327619.7 ms^2

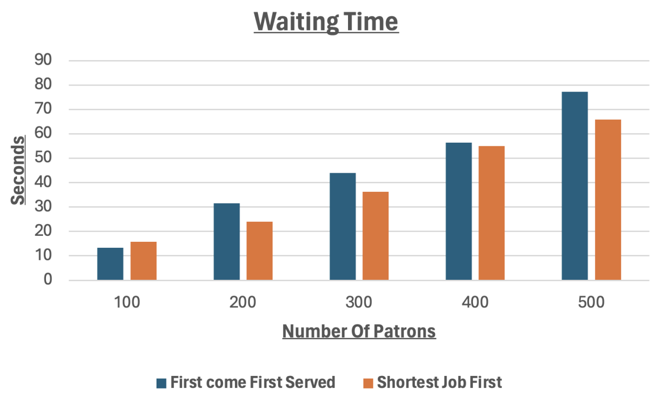
SJF Variance: 1635045534 ms^2

As can be seen above, the SJF is the better performing algorithm in all cases. This is because the SJF algorithm prioritises shorter jobs and as such ensures that less processes are waiting. This reduces the overall average of response time as if there is a section where the patrons suffer from a convoy effect of a large order, there are less patrons that are suffering from said effect which has less of an effect on the average. FCFS does not have this luxury and thus its response time metric suffers more from when a convoy effect occurs. As such, it can be concluded that another factor reducing the FCFS algorithm's performance in this metric is starvation. The variance is quite high yet again as different patrons have orders differing in size. That being said, SJF is more predictable than FCFS as it has a much lower variance.

Therefore SJF results in the best overall performance for the Response Time metric.

Waiting Time

Waiting time is the total amount of time that a process has been in the ready queue. To maximise performance, we wish to minimise this metric. In this program, this is represented by how long each drink of a patron remains in the queue while the barman is not working on it. This was calculated by summing all the times it would take the barman to make each drink and subtracting that value from the turnaround time, as that would be the amount of time that the barman was not working on the drinks. The average waiting time of patrons can be seen below:



FCFS Variance: 858834567.2 ms^2

SJF Variance: 1648692237 ms^2

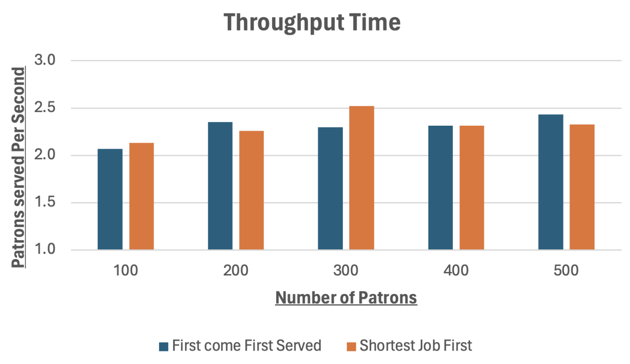
As can be seen, SJF is the worst performing algorithm for 100 patrons, yet as the number of patrons increases SJF becomes the better performing algorithm with regards to wait time. This is because patrons with smaller drink orders are placed in front of larger drink orders. This results in less average waiting time, as the patrons with smaller drink orders do not suffer from the convoy effect. From this, we can also conclude that the FCFS algorithm suffers from minor starvation as the convoy effect is present. There is a discrepancy at 400 patrons where the FCFS and SJF algorithms have very similar performance. This can be attributed to the random generation of drinks. The variance has the same trends as the previous metrics. This is due to the same reason where SJF reorders the queue while FCFS does not. As such, SJF is more predictable as it has the lower variance.

Therefore SJF results in the best overall performance for the waiting time metric.

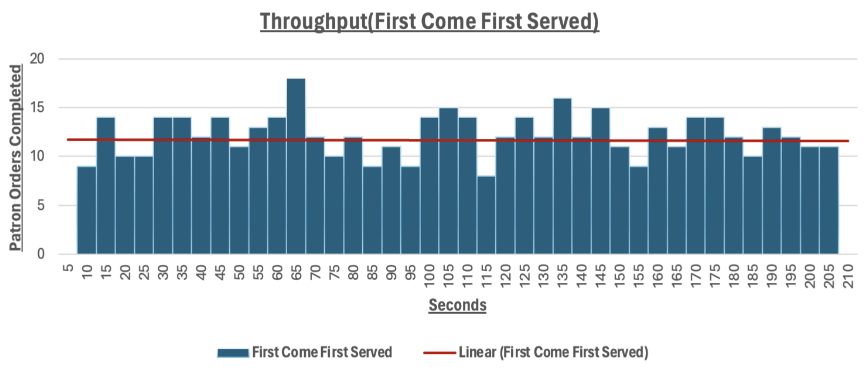
Throughput

Throughput is defined as the number of processes that complete their execution per unit time. Unlike the other metrics above, to maximise performance, we wish to maximise this metric. In this program, this is represented by the number of drink orders that the barman completes. This was calculated by using an atomic integer value that would be incremented every time that a patron received their full order of drinks. A TimeSchedulerExecutorService (which would perform its task every 5 seconds) would write that integer to an output file before setting the value back to 0. Three graphs were produced: 1 which was the average of the throughput times and 1 graph to monitor the throughput of each algorithm under the load of 500 patrons.

* Note that this method of calculating throughput does not account for any data that may be recorded in the last 4 seconds of the program's running

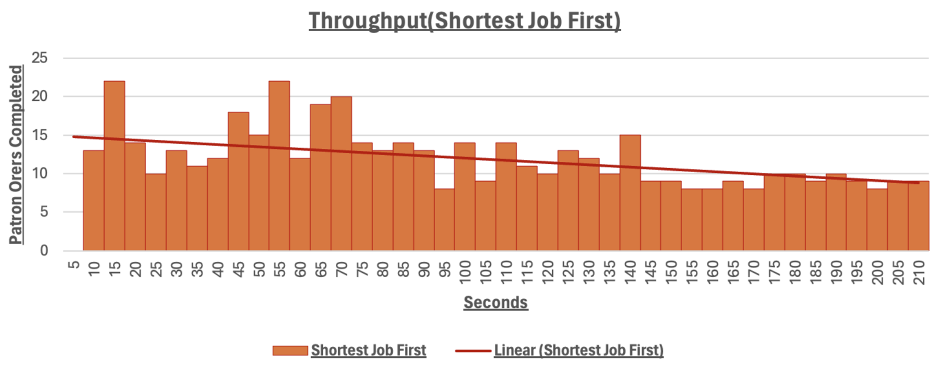


As can be seen by this graph, FCFS is the best performing algorithm. While SJF is the better performing of the 2 algorithms for 100 and 300 threads, this is due to the low number of patrons with large orders.



Average: 12.17

Variance: 13.20



Average: 11.64

Variance: 19.45

This trend is further illustrated in the 2 graphs above. SJF has a decreasing throughput time as the number of patrons increases while FCFS's throughput remains constant. This can be explained by how the SJF prioritises the shortest jobs and thus only large jobs (which require more time to prepare) are left. This is an example of how fairness benefits the FCFS algorithm as it ensures a relatively constant throughput, while the SJF has a decreasing linear trend as illustrated above. This decreasing trend shows that the SJF algorithm's performance continues to worsen as a result of an increase in patrons. The variance of the FCFS algorithm is also smaller than the SJF algorithm which means that it is more predictable under this metric. This is because the size of the orders remain relatively constant as FCFS is more fair and does not discriminate based on order size unlike SJF.

As such, it can be seen that FCFS is the superior algorithm with regards to this metric.

Conclusion

As can be seen by the above metrics, while the FCFS algorithm performed better in the throughput metric and is more fair, the SJF algorithm is the better performing algorithm overall as it has a better average waiting time, response time and turnaround time and is also more predictable in every metric save for throughput. More research may be required for a larger number of patrons to determine if the throughput has a larger impact, however this report can conclude that the SJF algorithm has the best overall performance.