# Bar Scheduling Simulation Project

Context and Motivation

The field of working frameworks broadly utilises CPU process scheduling to improve managing multiple processes. To develop an understanding of such scheduling algorithms serve as a vital instructive tool. This report showcase the simulations that models a active bar environment, representing the complex elements of process scheduling in a simple and quick situation.

Simulation Overview

In this simulations, a busy bar situation is utilised where "Andre the Barman" speaks to the CPU. Patrons arrive at arbitrary, each putting orders for between one to five drinks, associated to "employments" that the CPU must prepare. At first, Andre works using the First Come, First Served (FCFS) scheduling algorithm.

The primary objective of this assignment is to implement and evaluate an alternative scheduling strategy, Shortest Job First (SJF), and compare it to the existing FCFS strategy using a variety of performance metrics.

Objectives

1. Implement the SJF Algortihm
2. Evalute Performance metrics
3. Analyse the data
4. Create a visulation through graphs
5. Determine the better algortihm

Strategy

The project involves modifying the provided Java simulation package to include both FCFS and SJF scheduling algorithms. The code will include mechanisms to capture timing and performance data for each scheduling method. Five simulation runs will be conducted to ensure reliable data, which will be plotted and analysing the date using Excel.

Report Structure:

This report presents a comprehensive investigation of the bar scheduling to determine the best algorithm . It starts with a detailed discussion of the implementation, followed by an examination of the collected data, and concludes with an analysis of the FCFS and SJF scheduling algorithms. The findings from this report will guide recommendations on the optimal scheduling strategy.

Discussion on implementation:

* We used PriorityLinkedQueue for the SJF
* We created timestamps to measure the times acheievd as shown below.

Data collected:

The histograms(stacked and clustered) show the results and are further provided in the Graphs excel sheet.

A graph of different colored bars

Description automatically generated

Figure 1: The times achieved with 20 Patrons (SJF)

A graph of a number of people

Description automatically generated

Figure 2: The times achieved with 20 Patrons (FCFS)

The two graphs above show the relationship for twenty patrons between firstly SJF which shows that all values like FCFS are very close in time. However, if we look at the two graphs below, we see that the waiting time, turnaround time and response time are relatively low in most cases when looking at the SJF but in terms of FCFS it seems to have the same growth in waiting time, turn around time and response time.

Therefore, we can deduce that SJF offers significant improvements in efficiency for smaller batches of patrons.

A graph of different colored lines

Description automatically generated

Figure 3: The times achieved with 100 Patrons (SJF)

A graph of a number of people

Description automatically generated with medium confidence

Figure 4: The times achieved with 100 Patrons (FCFS)

We however can see from the averages that there is big difference between the values achieved by the SJK and FCFS for 20 Patrons and as the number of Patrons grew too 100 the algorithms seemed to produce similar results and that response time grew closer to FCFS. In both cases we do see that SJF is much better in time than FCFS.

A graph of different times of each algorithm

Description automatically generated

Figure 5: Average times of 20 Patrons

A graph of different times of each algorithm

Description automatically generated

Figure 6: Average times of 100 Patrons

Furthermore, we can show the trend of how SJF grows closer to FCFS.

A graph of different times of each algorithm

Description automatically generated

Figure 7: Average times of 40 Patrons

A graph of different times of each algorithm

Description automatically generated

Figure 8: Average times of 60 Patrons

Therefore, we can deduce that while SJF maintains a consistent advantage over FCFS in smaller groups, this advantage diminishes as the number of patrons increases.

We now need to look at the throughput which is a performance metric used to evaluate the efficiency of a system, particularly in the context of computing and data processing. It measures the rate at which a system (the barman) can process orders within a given period of time.

A graph of different colored bars

Description automatically generated with medium confidence

Figure 10: A histogram showing the difference between the SJF and FCFS throughput.

The graph above tells us that the time in seconds was higher of SJF on average in comparison to FCFS indicating that SJF, while faster in processing individual orders, does not scale as efficiently with higher numbers of patrons. This insight is critical for understanding the limitations of SJF in high-load scenarios.

With these graphs and analyses, the algorithm that should be used is SJF for smaller or moderate-sized groups of patrons due to its superior handling of job sizes and quicker response times. However, for larger groups, the differences between SJF and FCFS narrow, suggesting that the choice of algorithm might depend more on specific operational contexts rather than a one-size-fits-all solution.

This report has explored various facets of CPU scheduling by analogizing it to a bar scenario, providing valuable insights into the practical implications of different scheduling strategies.