CPU Scheduling Algorithms Evaluation

CS 3502: Project 2

Dang Tran
dtran42@students.kennesaw.edu - 001065099
CS 3502 Section 3

April 2025

Abstract

This project implements and compares multiple CPU scheduling algorithms. Extending from a given code base, this version includes two new algorithms: Shortest Remaining Time First (SRTF) and Multi-Level Feedback Queue (MLFQ). Performance metrics such as Average Waiting Time, Average Turnaround Time, CPU Utilization, and Throughput were evaluated using a base testing case across algorithms to recommend the most efficient strategy for OwlTech Systems.

1 Introduction

CPU scheduling is a fundamental concept in operating systems that determines which process will use the CPU next. Effective scheduling is critical to optimizing performance, responsiveness, and fairness among competing processes. This project aims to implement several scheduling algorithms and analyze their efficiency based on standard metrics.

2 Implementation Details

The provided CPU Simulator GUI starter code was extended by adding two new scheduling algorithms:

- **Shortest Remaining Time First (SRTF)**: A preemptive version of Shortest Job First (SJF), selecting the process with the least remaining burst time. At every time unit, the algorithm will select the process with the smallest remaining burst time.
- **Multi-Level Feedback Queue (MLFQ)**: Utilizes multiple queues with different priorities and time quanta, dynamically adjusting process priority based on CPU usage. Processes are demoted to lower-priority queues if they don't finish in time.

Users are asked to enter the total number of processes, followed by their arrival times and burst times. From then the algorithms will calculate the individual wait times and turnaround times for each process, as well as the average wait time and average turnaround time of all processes.

2.1 Changes to Starting Code Base

The new algorithms were added to the `Algorithms.cs` file. Additional GUI buttons were integrated into `CpuScheduler.cs` and `CpuScheduler.Designer.cs` to allow user selection of the SRTF and MLFQ methods.

2.2 Process Structure

Each process was defined with the following attributes:

- Arrival Time
- Burst Time
- Priority (for Priority Scheduling only)

3 Testing and Results

Testing was performed using a standardized set of 5 processes with set burst times. Two scenarios were tested:

3.1 Scenario 1: Different Arrival Times

Process	Arrival Time	Burst Time	Priority
P1	0	7	2
P2	2	4	1
P3	4	1	3
P4	5	10	2
P5	6	2	4

Algorithm	Avg Wait Time	Avg Turnaround Time	CPU Utilization	Throughput
FCFS	10.4s	11.8s	100%	0.208P/s
SJF	5s	10.2s	100%	0.208P/s
Priority	11.6s	16.4s	100%	0.208P/s
Round Robin	6s	11s	100%	0.208P/s
SRTF	3.6s	8.4s	100%	0.208P/s
MLFQ	7s	11.8s	100%	0.208P/s

3.2 Scenario 2: Arrival Time = 0

Process	Arrival Time	Burst Time	Priority
P1	0	7	2
P2	0	4	1
P3	0	1	3
P4	0	10	2
P5	0	2	4

Algorithm	Avg Wait Time	Avg Turnaround Time	CPU Utilization	Throughput
FCFS	10.4s	11.8s	100%	0.208P/s
SJF	5s	10.2s	100%	0.208P/s
Priority	11.6s	16.4s	100%	0.208P/s
Round Robin	9s	14s	100%	0.208P/s
SRTF	5s	9.8s	100%	0.208P/s
MLFQ	10.4s	15.2s	100%	0.208P/s

4 Analysis and Discussion

The results demonstrate the trade-offs between scheduling algorithms:

- **FCFS** - favors simplicity but may cause long waiting times (convoy effect).
- **SJF** - minimizes average waiting time but may suffer from starvation. Very close second to SRTF.
- **Priority Scheduling** - can be efficient but must handle starvation. Showed the slowest average waiting and turnaround times.
- **Round Robin** - improves responsiveness at the cost of higher context switching.
- **SRTF** - provides excellent waiting and turnaround times, but is more complex due to its preemptive nature. Overall, had the shortest wait and turnaround times throughout both tests.
- **MLFQ** - balances responsiveness and fairness, but requires careful tuning of queues and quantum times.

The two different test cases were used since the implementations of FCFS, SJF and Priority Algorithm do not take into account arrival times. Therefore, the algorithms were tested with different arrival times and all arrival times of 0. CPU utilization and throughput remained the same throughout. There was no rest time for the CPU.

5 Challenges and Lessons Learned

Challenges included:

- Acknowledging the current code base and understanding the techniques/design choices used
- Understanding and correctly implementing preemption for SRTF.
- Structuring the MLFQ with multiple queue levels and demotion rules.
- Integrating new algorithms cleanly into the GUI framework.
- FCFS, SJF and Priority Scheduling algorithms do not take into account arrival times in their implementation, which could change the wait and turn around times.

Lessons learned involved the importance of thorough testing and how small changes in scheduling strategy can drastically affect performance metrics. Extending from a given code base also proved to be a bit of a challenge, having to understand new technologies on top of documentation and all was challenging to take in.

6 Conclusion and Insights

This CPU scheduling simulation provided insights in how scheduling algorithms work and the differences between them. The project allowed for hands on experience in building these algorithms and testing how they work with different cases of processes. It also added the complexity of using a GUI, in which the new algorithms were added so they could function with the click of a button.

7 References

1 FrancisNweke, “GitHub - FrancisNweke/CPU-Simulator-GUI: A GUI program with a CPU simulator, QR code and barcode generator.,” GitHub, 2022. https://github.com/FrancisNweke/
Simulator-GUI (accessed Apr. 28, 2025).

2 GeeksForGeeks, “Multilevel Feedback Queue Scheduling (MLFQ) CPU Scheduling - GeeksforGeeks,” GeeksforGeeks, Aug. 16, 2019. https://www.geeksforgeeks.org/multilevel-feedback-queue-scheduling-mlfq-cpu-scheduling/

3 GeeksforGeeks, “Shortest Remaining Time First (Preemptive SJF) Scheduling Algorithm,” GeeksforGeeks, Jul. 08, 2017. https://www.geeksforgeeks.org/shortest-remaining-time-first-preemptive-sjf-scheduling-algorithm/