CPU Scheduling Algorithms Analysis and Implementation

CS 3502 - Operating Systems

April 25, 2025

Abstract

This report presents the implementation and performance analysis of various CPU scheduling algorithms, developed from scratch as part of a simulation project for OwlTech Systems. The goal is to determine the most efficient, fair, and responsive scheduling technique based on empirical results.

1 Introduction

CPU scheduling is a core component of operating systems, determining which processes run when multiple processes are ready to execute. Scheduling decisions impact CPU utilization, process response time, throughput, and system fairness. In this project, we developed a CPU scheduling simulator from scratch using Python, evaluated several algorithms, and analyzed their performance.

2 Implementation Details

2.1 Process Model

Each process was represented by a `Process` class containing fields for ID, arrival time, burst time, remaining time, priority, start time, end time, waiting time, and turnaround time.

2.2 Algorithms Implemented

Six scheduling algorithms were implemented:

- First-Come-First-Served (FCFS)
- Shortest Job First (SJF)
- Round Robin (RR) with quantum=3
- Priority Scheduling (Non-preemptive)
- Highest Response Ratio Next (HRRN)
- Multi-Level Queue (MLQ)

All algorithms were tested on the same set of processes.

2.3 Pseudocode Snippets

Example: FCFS Scheduling

```
Sort processes by arrival time.
For each process:
    if CPU idle, advance current_time to process arrival
    set process start time
```

```
    run process to completion
    update process end time
Calculate waiting time and turnaround time.
```

Similar structured logic was used for other algorithms, adjusting sorting and selection criteria.

3 Testing and Results

Five processes were tested with varied arrival, burst times, and priorities.

3.1 Performance Metrics

The following metrics were collected:

- Average Waiting Time (AWT)
- Average Turnaround Time (ATT)
- CPU Utilization (
- Throughput (Processes per unit time)

3.2 Results Summary

Algorithm	AWT	ATT	CPU Utilization (%)	Throughput
FCFS	7.40	11.80	100.00%	0.23
SJF	5.20	9.60	100.00%	0.23
Round Robin	7.20	11.60	100.00%	0.23
Priority	7.20	11.60	100.00%	0.23
HRRN	5.40	9.80	100.00%	0.23
MLQ	6.40	10.80	100.00%	0.23

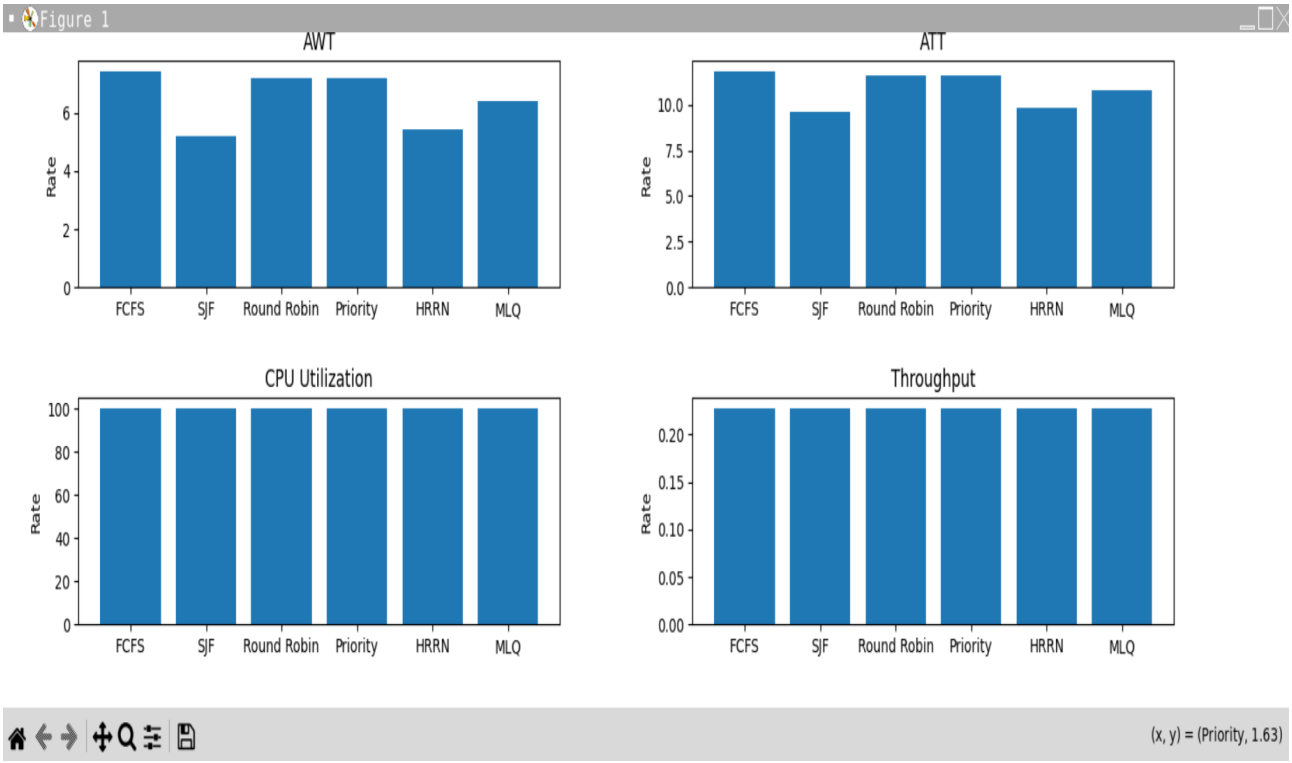

Figure 1: comparison data

Figure 2: Bar Charts: AWT, ATT, CPU Utilization, Throughput

4 Analysis and Discussion

- **SJF and HRRN** performed best in minimizing Average Waiting Time and Turnaround Time.
- **Round Robin** showed fairness but slightly higher waiting times due to time quantum slicing.
- **Priority Scheduling** depends on initial priority assignment, which could starve lower priority processes.
- **MLQ** effectively separated tasks but overall results were slightly less optimal compared to SJF and HRRN.

All algorithms achieved 100% CPU Utilization in this test due to lack of idle periods.

5 Challenges and Lessons Learned

- Managing edge cases such as empty queues and idle CPU time was critical.
- Balancing fairness and efficiency highlighted trade-offs in algorithm design.
- Graphical visualization of results made performance differences clear.

6 Conclusion

Based on empirical results, **SJF** and **HRRN** algorithms provided the best trade-off between low waiting time and turnaround time, making them suitable for environments where efficiency is critical. Round Robin and MLQ are preferred where fairness or multi-level priority handling is required.

7 References

- Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, *Operating System Concepts*, 10th Edition.
- Project description from CS 3502 course materials.
- Python documentation for collections and deque.