Report on the Implementation and Results of Multilevel Queue Scheduling Algorithm

K.Keeththigan — 2023/cs/093 — 23000937

February 24, 2025

1 Introduction

This report provides an overview of a simulation program that implements the Multilevel Queue Scheduling (MLQ) algorithm. The program simulates scheduling processes using four distinct algorithms for each queue:

- Round Robin (RR) for queue 0
- Shortest Job First (SJF) for queues 1 and 2
- First In First Out (FIFO) for queue 3

The purpose of this program is to demonstrate the functioning of each scheduling algorithm in a multilevel queue environment, calculate the average waiting time and turnaround time for each queue, and evaluate the pros and cons of these algorithms.

2 Implementation Details

The simulation divides processes into four queues, each implementing a different scheduling algorithm.

2.1 Process Class

The `Process` class is used to represent each process. The class contains:

- `id`: A unique identifier for the process.
- `burstTime`: The time required for the process to execute.
- `remainingTime`: The time left for the process to complete execution.
- `waitingTime`: The time the process has been waiting in the queue.
- `turnaroundTime`: The total time taken from the process arrival to its completion.
- `priority`: The queue to which the process belongs.

2.2 Scheduling Algorithms

Round Robin (RR): Used in queue 0, RR gives each process a fixed time slice, and processes are scheduled in a circular manner. If a process doesn't finish within its time slice, it goes back to the queue for the next round.

Shortest Job First (SJF): Applied to queues 1 and 2, SJF schedules processes based on their burst times. The process with the shortest burst time gets executed first, reducing the average waiting time for shorter tasks.

First In First Out (FIFO): Used in queue 3, FIFO schedules processes in the order they arrive, without considering their burst times. It's a simple, fair scheduling approach but can result in higher waiting times for processes with larger burst times.

2.3 Multilevel Queue Scheduling

The `MultilevelQueueScheduling` class coordinates the scheduling, applying the appropriate algorithm to each queue and ensuring that processes are executed correctly. The program checks each queue, runs the processes according to their scheduling algorithm, and calculates their waiting and turnaround times.

3 Example Input and Output

3.1 Example Input

```
Enter the number of processes: 5
Enter queue level (0-3) for process 1: 0
Process 1 - Burst Time: 15, Queue Level: 0
Enter queue level (0-3) for process 2: 1
Process 2 - Burst Time: 5, Queue Level: 1
Enter queue level (0-3) for process 3: 2
Process 3 - Burst Time: 10, Queue Level: 2
Enter queue level (0-3) for process 4: 3
Process 4 - Burst Time: 20, Queue Level: 3
Enter queue level (0-3) for process 5: 0
Process 5 - Burst Time: 8, Queue Level: 0
```

3.2 Example Output

```
Queue 0 - Average Waiting Time: 10.5, Average Turnaround Time: 18.5
Queue 1 - Average Waiting Time: 0.0, Average Turnaround Time: 5.0
Queue 2 - Average Waiting Time: 5.0, Average Turnaround Time: 15.0
Queue 3 - Average Waiting Time: 10.0, Average Turnaround Time: 30.0
```

In this example:

- Processes 1 and 5 are scheduled using Round Robin (RR), which distributes CPU time fairly among the processes.
- Process 2 uses Shortest Job First (SJF), which executes it first because it has the smallest burst time.

- Process 3 is also scheduled with SJF, as it has a smaller burst time compared to others in the same queue.
- Process 4 is scheduled using FIFO, as it belongs to queue 3.

4 Results

The results show the average waiting and turnaround times for each queue and highlight the strengths and weaknesses of each scheduling algorithm:

4.1 Queue 0 (Round Robin)

Pros: Round Robin ensures that no process is left waiting too long by allocating equal time slices to each process, making it ideal for systems where all processes should have equal access to the CPU.

Cons: If processes have significantly different burst times, this can result in inefficient use of CPU time. Processes with larger burst times may end up being delayed unnecessarily.

4.2 Queue 1 and Queue 2 (Shortest Job First)

Pros: SJF minimizes the average waiting time by always selecting the shortest job. It's efficient when processes have known burst times.

Cons: One drawback is the convoy effect, where longer processes have to wait for shorter ones to complete. Additionally, SJF requires knowledge of the burst time beforehand, which might not always be feasible.

4.3 Queue 3 (FIFO)

Pros: FIFO is straightforward and ensures fairness, as processes are executed in the order they arrive without preemption.

Cons: It doesn't take burst times into account, which can lead to poor performance if longer processes arrive first, causing high waiting times for subsequent processes.

5 Conclusion

The Multilevel Queue Scheduling simulation effectively demonstrates how different scheduling algorithms can be used to manage processes with varying burst times. Round Robin offers fairness, Shortest Job First minimizes waiting times for shorter processes, and First In First Out provides a simple, fair scheduling approach. However, each algorithm comes with its own set of strengths and weaknesses, depending on the nature of the processes being scheduled.

Each algorithm was designed to handle processes in a specific way, and their performance varied based on the processes assigned to them. Round Robin is best suited for time-sharing systems, SJF excels when burst times are known and predictable, and FIFO offers a simple approach that works well in many cases.