CPU Scheduler

1. Introduction

In this project, I have implemented various CPU scheduling algorithms used in operating systems, along with a user-friendly frontend interface. The goal was to provide a comprehensive solution for visualizing and understanding different CPU scheduling techniques, including First-Come, First-Served (FCFS), Shortest Job First (SJF), Priority Scheduling, Round Robin (RR), and an Improved Round Robin (IRR) algorithm.

2. Project Details

a. Backend

The backend is implemented in C++, handling the core logic and computations for CPU scheduling algorithms. It defines data structures like Process and Output to represent processes and scheduling results, respectively.

Algorithm Implementations: The backend provides implementations for various CPU scheduling algorithms, each following a specific logic to determine the execution order of processes:

- **First-Come, First-Served (FCFS):** This algorithm schedules processes based on their arrival time. Processes are sorted by arrival time, and each process is executed in that order until completion. The fcfs() function implements this algorithm, iterating through the sorted processes and updating the completion time and Gantt chart accordingly.
- **Shortest Job First (SJF):** This algorithm schedules processes based on their burst time (CPU time required). Processes are sorted by arrival time, and at each time instance, the process with the shortest remaining burst time is executed. The sjf() function maintains a ready queue and selects the process with the shortest job from the queue for execution.
- **Priority Scheduling:** This algorithm schedules processes based on their assigned priority. Processes are sorted by arrival time, and at each time instance, the process with the highest priority is executed. The priority() function implements this algorithm by maintaining a ready queue and selecting the highest priority process from the queue.
- **Round Robin (RR):** This algorithm schedules processes by assigning a time quantum (specified by the user) to each process. The CPU cycles between processes, executing each for the time quantum before moving to the next process. The rr() function implements the Round Robin algorithm using a ready queue and updating the remaining time for each process.
- **Improved Round Robin (IRR):** This is an improved version of the Round Robin algorithm. It starts by executing the Round Robin algorithm for one round, and then it sorts the remaining processes based on their remaining burst time and executes them using the Shortest Job First algorithm. The irr() function implements this algorithm by combining the Round Robin and Shortest Job First approaches.

The backend also includes helper functions to calculate metrics like average waiting time, average turnaround time, and CPU utilization.

To interact with the backend, first compile the file using the command:

```
g++ -o backend/scheduler backend/algorithms/scheduler.cpp
```

b. Frontend

The frontend is developed using HTML, CSS, and JavaScript, providing a user-friendly interface. Users can input the number of processes, along with their arrival time, burst time, and priority (if applicable). They can also select the desired scheduling algorithm from a dropdown or set of buttons. After the backend computations, the frontend displays the results, including process details, average waiting time, average turnaround time, CPU utilization, and a graphical representation of the Gantt chart.

To set up the frontend, navigate to the project directory and run the following commands:

```
npm install
```

Then, start the application using:

```
npm start
```

c. Integration

To integrate the backend (C++) and frontend (JavaScript), I utilized Node.js with C++ addons. This allowed the frontend to make HTTP requests to the server, which in turn executed the C++ code and returned the results to the frontend for visualization.

3. Features and Additional Creativity

- **Interactive Gantt Chart**: The frontend provides an interactive Gantt chart visualization, allowing users to hover over or click on process blocks to display additional information.

- **Algorithm Comparison**: Users can select multiple algorithms and compare their performance side-by-side.

- **Best Algorithm Recommendation**: The project analyzes the results of all implemented algorithms and recommends the best algorithm based on the lowest average waiting time and average turnaround time.

- **Process Prioritization**: For the Priority Scheduling algorithm, users can dynamically adjust the priority of processes and observe the impact on scheduling order and performance metrics.

4. Optimal Scheduling Algorithm

Among the implemented scheduling algorithms, Round Robin (RR) is considered the most optimal and widely used in modern time-shared computer systems. However, each algorithm has its own advantages and disadvantages, which are discussed below:

First-Come, First-Served (FCFS)

• Advantage: Simple and easy to implement, doesn't lead to starvation (indefinite waiting for a process).

• Disadvantage: Non-preemptive (processes are not interrupted once started), average waiting time is higher compared to other algorithms.

Shortest Job First (SJF)

• Advantage: More optimal than FCFS, can be preemptive (processes can be interrupted if a shorter job arrives).

• Disadvantage: Can lead to infinite blocking or starvation (longer processes may never get a chance to execute).

Priority Scheduling

• Advantage: Allows prioritization of important processes, ensuring they get executed sooner.

• Disadvantage: Can lead to starvation of lower-priority processes if higher-priority processes keep arriving.

Round Robin (RR)

• Advantage: Most efficient among all algorithms, used in many modern computers. Preemptive in nature, hence doesn't lead to starvation.

• Disadvantage: Overhead of context switching (switching between processes can be computationally expensive).

Improved Round Robin (IRR)

• Advantage: Combines the benefits of Round Robin and Shortest Job First algorithms, minimizing waiting time and turnaround time while avoiding starvation.

• Disadvantage: Slightly more complex to implement than the basic Round Robin algorithm.

5. Summary and Further Improvements

The project successfully implemented various CPU scheduling algorithms, providing a user-friendly interface for visualization and understanding their behavior. The integration of the backend and frontend components allowed for efficient computations and interactive visualizations.