Github link: **https://github.com/lasyaE04/SYSC4001_A3_P1**

Introduction

This report performs a comprehensive analysis on the three CPU scheduling algorithms implemented in the simulator: External Priorities (EP), Round Robin (RR) and External Priorities + Round Robin (EP-RR). External Priorities is a non-preemptive scheduler where processes with lower PID values have higher priority. Once a process starts running on the CPU, it continues until completion or when an I/O operation is needed. Round Robin is a pre-emptive scheduler where each process receives a time quantum of 100 ms. When the time slice expires, the process is moved to the ready queue. This allows for the CPU to be shared more fairly amongst all the processes. External Priorities + Round Robin scheduler combines both approaches, resulting in pre-emption based on PID number while enforcing the time quantum of 100 ms for each process with the same priority.

Each of the three schedulers were used to run 20 test cases with mostly I/O bound processes, mostly CPU-bound processes, and processes with a mix of I/O and CPU bursts. To compare the schedulers under these conditions, four metrics were calculated:

Throughput: number of processes completed per unit time

Average Wait Time: the time a process spends in the ready queue

Average Turnaround Time: the total time between a process's arrival to its completion

Average Response Time: the time between a process's arrival and first execution

In addition, the use of memory was also recorded in the simulation to analyze the total memory usage, the partition allocation efficiency and the impact of memory blocking.

Analysis

Table 1: Performance Metrics for Various Schedulers

Metric	EP	RR	EP_RR	Best Scheduler
Average Throughput (processes/sec)	0.0082	0.0079	0.0085	EP-RR
Average Wait Time (ms)	89.3	142.7	76.4	EP-RR
Average Turnaround Time (ms)	245.8	298.2	232.1	EP-RR
Average Response Time (ms)	12.4	8.6	9.2	RR
Number of Memory Blocking Events	18	18	15	EP-RR

Based on Table 1, EP-RR performs better in all metrics except average response time due to it prioritizing high-priority processes, while fairly distributing the CPU time among processes. RR has a faster average response time than the two other schedulers because every process receives an equal timeslice and a new process is set to run at the end of that timeslice. This reduces overall latency by ensuring that high priority tasks do not occupy the CPU till completion, leading to CPU starvation for lower priority processes.

Table 2: Performance by Workload Type for Various Schedulers

Workload Type	Test Cases	EP Average Turnaround Time (ms)	RR Average Turnaround Time (ms)	EP-RR Average Turnaround Time (ms)	Best Scheduler
CPU-bound	1, 3, 5, 6, 11, 12, 15, 17, 19	198.4	287.3	185.2	EP-RR
I/O-Bound	2, 4, 13, 14, 18, 20	312.8	265.4	289.6	RR
Mix of I/O and CPU bursts	7, 8, 9, 10, 16	224.6	342.8	221.8	EP-RR

For CPU-bound processes, EP-RR results in a better turnaround time as high-priority processes can complete quickly without waiting for lower priority processes to run for their time quantum. EP performs similarly to EP-RR but has a disadvantage where lower priority processes block higher priority processes arriving due to the lack of a timeslice (the lower priority process runs to completion or until it needs to do I/O regardless of whether a higher priority process arrives or not). RR performs the worst as it forces all processes to wait for their timeslice, resulting in high turnaround times especially for processes arriving later.

I/O bound processes spend a significant amount of time in the wait queue, which increases the overall turnaround time. RR performs the best because as one process is doing I/O, all others can make progress by taking their turn using the CPU. EP performs the worst because lower priority processes must wait longer in the ready queue before their CPU burst, which delays their next I/O operation. This is mitigated slightly in EP-RR, which performs better than EP due to its use of timeslices.

For processes with a mix of I/O and CPU bursts, EP-RR performs the best because it lets high priority CPU bound processes complete quickly while also allowing for I/O bound processes to run on the CPU when they are ready, so they can do their next I/O with less delay. RR performs the worst because it treats all processes equally and CPU-bound processes are repeatedly interrupted and must wait their time quantum to reach completion. EP also performs worse than EP-RR as it is non-preemptive and can cause starvation for lower-priority processes.

Memory Utilization Analysis

Table 3: Memory Utilization Metrics for Various Schedulers

Metric	EP	RR	EP_RR
Average Memory Usage (MB)	68.4	68.4	71.2
Peak Memory Usage (MB)	100	100	100
Number of Memory Blocking Events	18	18	15

Average Time Blocked (ms)	42.3	42.3	35.1
Memory Turnover Rate (processes/ms)	0.0089	0.0086	0.0094

According to Table 3, EP-RR has the best memory turnover rate, the rate at which the memory partitions are freed by completed processes. That is because EP-RR results in higher throughput and completes high priority processes faster, freeing memory partitions sooner for arriving processes. Therefore, it also has less memory blocking events than the other two schedulers.

Conclusion

The results of the simulation show that EP-RR achieves the best performance overall. However, to determine the best scheduler for a particular application, the system and the type of workload must be considered.