Report

SYSC 4001A (L3)
Student 1: Eshal Kashif (101297950)
Student 2: Emma Wong (101297761)

GitHub Link: https://github.com/EmmaWong8/SYSC4001_A3_P1

Introduction

This report analyzes the behavior of three scheduling algorithms:External Priority (EP), External Priority with Round Robin (EP_RR), and Round Robin (RR). This is done so by using twenty simulation scenarios where each scenario was executed through a scheduling simulator that models process arrivals, CPU bursts, I/O events, context switches, and quantum expiration. For each simulation, the following metrics were collected and analyzed: throughput, average waiting time, average turnaround time, and average response time. The data for these metrics can be found in the CSV files on git: here. The quantum was set to 100 ms for all tests except Test Case 13, where it was reduced to 25 ms to study quantum vs I/O interaction. The goal of this report is to compare and interpret each scheduler's performance under CPU-bound, I/O-bound, and mixed workloads.

Overview of Scheduler Behaviour

External Priority (EP)
EP is a non preemptive priority scheduler. Once a process begins running, it continues until it terminates or performs an I/O operation. The priority is determined by PID, where smaller PID = higher priority. Since EP does not preempt on arrival, if a lower priority process begins running before higher priority processes arrive, it will hold the CPU until termination or I/O. In scenarios where processes arrive in descending priority order (e.g., PID 10 before PID 1), this makes EP behave similarly to FCFS, ignoring the priority mechanism entirely. Across many of the CPU-bound scenarios, EP produced the largest waiting times for higher priority jobs that arrived later and resulted in the highest waiting and turnaround times when long CPU bound jobs delayed the start of other processes.

External Priority with Round Robin (EP_RR)
EP_RR adds preemption to EP. A process is preempted if a higher priority process arrives or if its quantum expires. This scheduler favors high priority tasks so whenever a higher priority process becomes READY, it immediately takes the CPU. In scenarios mixing long and short jobs, EP_RR significantly reduces the response time of high priority tasks but introduces more context switching. Throughput remained similar to EP and RR, so the main differences appeared in waiting time and turnaround time rather than completion rate.

Round Robin (RR)
RR ignores priority completely and cycles through processes in the ready queue with a fixed quantum. Since each READY process receives CPU time in equal slices, RR is typically the most fair scheduler. It avoids starvation and provides consistently low response time for all processes. However, in workloads where high priority, short tasks are expected to finish quickly, RR may delay them unnecessarily. In heavily I/O-bound scenarios where quantum rarely expires, RR begins to behave similarly to EP_RR, but without any priority bias.

CPU-Bound Scenarios (Tests: 3, 5, 6, 9, 10, 12, 16, 20)

Throughput remained identical across all three schedulers, so analysis focuses on waiting, turnaround, and response time. The CPU-bound averages from the metric tables are:

- **EP**: Waiting =169.38 ms, Turnaround = 374.88 ms, Response = 180.17 ms
- **EP_RR**: Waiting = 128.46 ms, Turnaround = 340.83 ms, Response = 49.81 ms
- **RR**: Waiting = 180.17 ms, Turnaround = 389.92 ms, Response = 73.92 ms

EP's non preemptiveness causes higher priority processes to wait when long CPU-bound processes execute first. Tests 5 and 6 demonstrate this: PID 10 ran its full burst before PID 1 or PID 0 received CPU time. This behavior raises waiting and turnaround times for high priority processes.

EP_RR gives the best performance in CPU-bound workloads. In tests 9 and 10, EP_RR immediately switched to PID 1 and PID 0 when they arrived, reducing waiting and turnaround significantly compared to EP and RR. This is consistent with EP_RR's lowest averages (128.46 ms waiting, 340.83 ms turnaround, and 49.81 ms response).

RR shows fair rotation but the highest waiting and turnaround times in CPU-bound workloads. In test 12, RR alternated between PID 1 and PID 10 every quantum, leading to larger turnaround values than EP_RR. The RR averages (180.17 ms waiting, 389.92 ms turnaround) confirm its weaker performance in this category.

I/O-Bound Scenarios (Tests: 2, 4, 7, 8, 11, 14, 18)

I/O-bound workloads involve frequent blocking, which reduces the effect of the CPU scheduling algorithm. The metric tables results:

- **EP**: Waiting = 56.71 ms, Turnaround = 178.57 ms, Response = 29.78 ms
- **EP_RR**: Waiting = 51.0 ms, Turnaround = 172.85 ms, Response = 19.28 ms
- **RR**: Waiting = 56.71 ms , Turnaround = 178.57 ms, Response = 22.64 ms

We can see that EP_RR has the lowest averages across I/O-bound tests, EP and RR have nearly identical averages, slightly worse than EP_RR.
For example:

- Test 7: Both processes perform I/O every 2 ms. The quantum is never reached, so EP, EP_RR, and RR produced identical behavior and identical metrics.
- Test 8: CPU bursts are long enough to reach the quantum. EP_RR and RR both preempt once per 100 ms burst before the I/O occurs at 150 ms. EP completes in 350 ms, RR in 350 ms, and EP_RR in 353 ms. This 3 ms difference is the result of a single process left over to run and while it is performing I/O, the CPU is left idle.
- Test 11: The simulator correctly prioritizes process termination over I/O when both occur at the same time, and all three algorithms behave similarly.

Because device wait time dominates these workloads, EP, EP_RR, and RR all complete processes at nearly the same times. The differences in waiting and turnaround time are small, with EP_RR slightly better.

Mixed CPU & I/O Scenarios (Tests: 1, 13, 15, 17, 19)

Mixed workloads combine CPU bursts and I/O events, making scheduler behavior more varied.
Metric table averages for mixed workloads:

- **EP:** Waiting = 226.75 ms, Turnaround = 412.25 ms, Response = 137.17 ms
- **EP_RR:** Waiting = 192.19 ms, Turnaround = 379.0 ms, Response = 112.474 ms
- **RR:** Waiting = 298.38 ms, Turnaround = 484.25 ms, Response = 57.83 ms

RR shows lower average response time in mixed workloads because it provides rapid, fair rotation to all processes regardless of priority. While EP_RR gives excellent response times to high priority processes, low priority processes may wait much longer for their first CPU time, thus increasing the overall average. Compared to CPU-bound workloads, EP_RR's immediate preemption for high priority arrivals dominates the average and results in better overall response times.

EP_RR performs best, with the lowest waiting and turnaround times. Here are some test case examples:

- Test 15: Three processes with different I/O frequencies. EP and EP_RR respect priority, giving PID 0 more CPU access. RR cycles evenly. EP_RR shows the lowest turnaround time in this test, matching its overall average.
- Test 17: A late high priority arrival. EP does not preempt PID 10, causing PID 1 to wait roughly 150 ms before running. EP_RR preempts immediately, giving PID 1 minimal response time. RR places PID 1 into the rotation but does not give priority based preemption.

- Test 19: A complex five process workload. EP_RR aggressively reorders processes after each I/O event based on priority. RR remains uniform. The metric tables confirm wide spreads in EP and EP_RR but uniform values in RR.
- In Test 13, the reduced quantum (25 ms) highlights that I/O takes priority over quantum expiration in both EP_RR and RR. EP behaves as expected for a non preemptive scheduler. This confirms the simulator's correct handling of simultaneous I/O and quantum boundaries.

Conclusion

Across all twenty scenarios, throughput was effectively identical for EP, EP_RR, and RR. Meaningful differences were found in waiting, turnaround, and response times.

- EP_RR consistently performed best, with the lowest waiting and turnaround times in CPU-bound and mixed workloads and slightly better performance in I/O-bound workloads.
- EP performed in the middle: worse than EP_RR but better than RR in CPU-bound and mixed scenarios.
- RR produced the highest waiting and turnaround times in CPU-bound and mixed tests, but had the most uniform process to process behavior and often lower response times due to rapid rotation.

The response time differences highlight an important trade-off: EP_RR optimizes for high priority task responsiveness, which benefits CPU-bound workloads where priority enforcement matters most. However, in mixed workloads with diverse priorities, RR's fairness approach provides more consistent response times across all processes. This demonstrates that EP_RR's "better" performance depends on whether minimizing high priority response time or maintaining fair average response time is the primary goal.

From our results we can see that EP with RR is generally better than using either EP or RR alone, because it leverages the strengths of both approaches. However, no single scheduler is optimal for all situations. EP_RR is best when priority enforcement and minimal waiting time are required. EP is suitable for simple, predictable, non preemptive environments. RR is most appropriate when fairness is more important than minimizing completion time.