

Simulation Analysis.

This report analyzes the performance of the following three CPU scheduling algorithms under three different workload conditions:

- External Priorities
- Round Robin
- External Priorities with Round Robin

The analysis is based on execution logs from which we computed key performance metrics, such as throughput, waiting time, turnaround time, and response time.

Test Scenarios

We had 20 different test cases, each process entry includes PID, memory size, arrival time, total processing time, I/O frequency, I/O duration, and priority. Tests were categorized as:

- CPU-bound: Long CPU bursts, infrequent I/O.
- I/O-bound: Frequent I/O operations, short CPU bursts.
- Mixed: Balanced characteristics.

Performance Metrics

For each scheduler on each test the following metrics were calculated:

- Throughput: Processes completed per millisecond.
- Average Wait Time: Mean duration in READY state.
- Average Turnaround Time: Completion time minus arrival time.
- Average Response Time: First execution minus arrival time

Results and Analysis

Average Throughput:

- EP: 0.031 process/ms
- RR: 0.029 process/ms
- EP_RR: 0.030 process/ms

Average Wait Time:

- EP: 42.7 ms
- RR: 38.2 ms
- EP_RR: 40.1 ms

Average Turnaround Time:

- EP: 218.4 ms
- RR: 225.1 ms
- EP_RR: 221.8 ms

Average Response Time:

- EP: 18.3 ms
- RR: 12.6 ms
- EP_RR: 15.4 ms

Workload-Specific Performance

CPU-bound Workloads: EP was much better for processes that require extensive CPU use, achieving faster turnaround times than RR. It being non-preemptive allowed high-priority processes to complete long CPU bursts without interruption.

I/O-bound Workloads: RR did better than EP for I/O-intensive workloads, reducing average wait significantly. This is likely due to RR's ability to handle more frequent I/O interactions.

Algorithm Behavior Analysis

EP Scheduler Characteristics:

Strengths: Minimal overhead, optimal for jobs requiring strict priority acceptance.

Weaknesses: High chance of starvation observed through large wait times for lower priority processes.

Best Performance: CPU bound processes, with a strict priority order.

RR Scheduler Characteristics:

Strengths: Excellent response time, had consistent performance across processes, smallest average wait time between other schedulers.

Weaknesses: Much higher amount of context switches, compared to EP leading to higher turnaround time.

Best Performance: Pure I/O-bound processes.

EP_RR Scheduler Characteristics:

Strengths: Balanced across all metrics, not necessarily the best or the worst on any aspect.

Weaknesses: Much higher overhead compared to just EP, and longer average wait time than RR.

Best Performance: Any test with mixed characteristics, not strictly I/O or CPU bound.

Key Findings

A clear trade-off exists between throughput and response time. EP achieves highest throughput (0.031 proc/ms) but has the worst response time (18.3 ms). RR provides the best response time (12.6 ms) but with a lower throughput. EP_RR is fairly average as it balances both aspects.

The 100ms quantum proved optimal for the test mix. Analysis showed that quantum size directly affects frequency of context switching. This is expected, as the smaller the quantum is the more often the CPU switches between jobs, leading to higher context switches.

RR demonstrated superior I/O handling, completing I/O-bound tests a lot faster than EP. The time-sharing approach allowed overlapping of CPU execution with I/O operations from other processes.

Conclusion

This analysis demonstrates that scheduling algorithm choice significantly impacts overall system performance metrics. EP provides the highest throughput for CPU-intensive workloads but suffers in response time. RR offers excellent responsiveness for interactive applications with frequent I/O. EP_RR provides the most balanced outcomes suitable for mixed workloads. The 100ms quantum in RR and EP_RR represents an effective compromise between responsiveness and overhead. For the tested scenarios, EP_RR delivered the most consistent performance across all workload types, making it a suitable choice for systems with varying workload characteristics, or when type of workload might not be known.

Part 1 Repo: https://github.com/JesseHanda/SYSC4001_A3_P1.git

Part 2 Repo: https://github.com/JesseHanda/SYSC4001_A3P2.git