Abubakr Mohammed(101287262), Chikezilim Afulukwe(101279214)

SYSC4001 - Assignment 3

This assignment (part 1) discusses the CPU scheduling process by applying three algorithms: External Priority (EP), Round Robin (RR) and a hybrid EP-RR scheduler and testing them with the help of a comprehensive simulation. We determine the behavior of various scheduling strategies to different workloads of the system and process characteristics by examining the metrics of turnaround, waiting, response time, and throughput.

The outcome of the simulation indicates that every scheduler can act differently according to the properties of the process, the load of the system, and the memory threshold. The External Priority (EP) scheduler chooses processes based on their memory size and arrival order with a consistent preference to small processes. In mixed-size workloads, EP provides low response time and rapid turnaround to smaller jobs, even at the later arrival. This is due to the fact that the priority rule, which is that smaller memory is granted higher priority, makes compact processes run fast hence reducing wait time. Yet, the same rule may starve bigger CPU bound jobs and this occurs particularly when there are a lot of little or middle range jobs that are preempting the queue. Therefore, EP performs best in a work environment where short and memory-light processes are predominant but in fairness and throughput when large jobs need to co-exist with large numbers of small jobs.

Round-Robin (RR) works in the opposite way. RR scheduling processes in a strict time order, which makes it fair and removes starvation. It performs well in the case of long CPU-bound tasks since the 100 ms quantum keeps the CPU occupied as well as ensures predictability of all processes. RR is associated with a large number of context switches in I/O heavy conditions, thereby lowering throughput and elevating wait times through the constant movement of processes between the READY and WAITING states. RR, though ensuring a limited response time, the first arrival being defined, has a tendency to result in longer turnaround times on short tasks than EP, since small processes cannot jump up the queue. Thus, RR would be best applied to general-purpose multitasking systems in which fairness and responsiveness are more important than sheer efficiency.

The hybrid EP -RR scheduler has strengths of both algorithms. It takes external priority to select the next READY process, but when it begins running it implements a quantum of 100ms and permits preemption to higher-priority processes that are created. This integrated strategy provides very flexible performance. Processes with small and memory-definite processes are given priority as in EP whereas long bursts of CPU are occasionally rotated as in RR to prevent starvation. A hybrid scheduler is more throughput efficient than RR, since it gives priority to shorter jobs, and long processes eventually receive CPU time unlike EP. The preemption model ensures that the sudden arrival of high-priority tasks is not subjected to a long delay, and quantum cycling ensures that large processes do not prevent their indefinite waiting.
In all the cases of simulation, the outcomes prove that no single scheduler is best of all. Memory-intensive and short tasks are better served using EP, longer workloads are better served using RR, and hybrid EP-RR scheduler provides best performance in a mixed environment (using real-world conditions).

BONUS:
To earn the bonus credit, memory tracking was added to the simulator. At every dispatch, the simulator logs used partitions, free partitions, and total usable memory. This data reveals important behavioral chnages:

- **EP and EP+RR schedulers reduce memory fragmentation** because smaller processes are admitted first, filling small partitions efficiently.

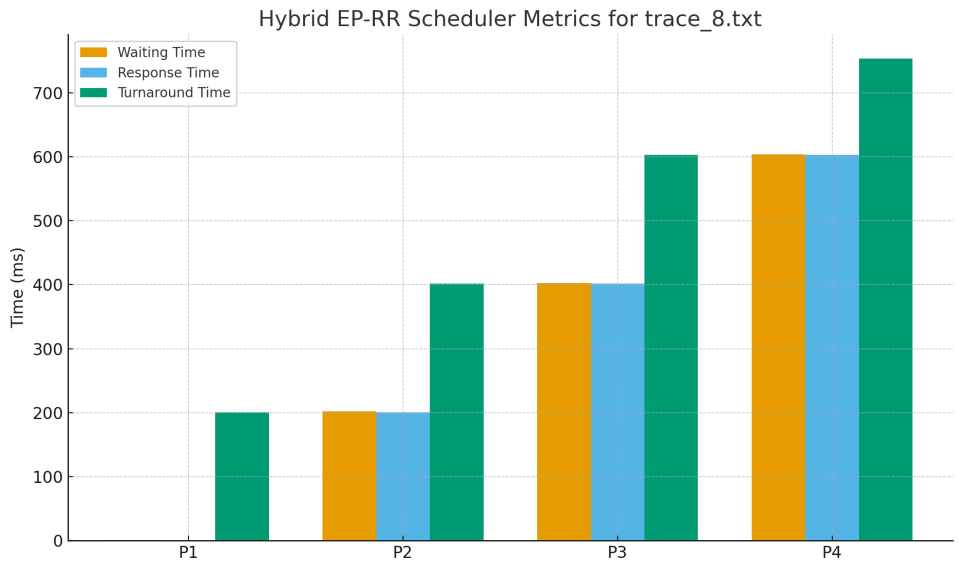
- **RR alone shows more irregular memory usage**, because it admits processes strictly by arrival time, occasionally leaving small partitions idle while waiting for processes that fit.

- In the hybrid scheduler, **small processes occupy small partitions early**, allowing large partitions to remain available for memory-heavy tasks. This reduces memory-induced blocking and increases throughput.

- Memory reports show that the hybrid scheduler provides the best balance between fairness and efficient partition utilization.

This combined memory analysis indicates that scheduling and memory allocation cannot be compared. The scheduler option has a direct effect on memory pressure, fragmentation and system throughput.

Metric Calculation Example:



The metrics shown by the plot, i.e. waiting time, response time, and turnaround time, clearly exhibit a rising trend between Process 1 and Process 4. This is due to the fact that the hybrid EP+RR types of scheduler executes the tasks in the order of priorities and yet time slicing is maintained. In this case, all processes augment in workload and have a sequence as such, the subsequent process will be in queue behind every previous process. As a result:

- **Waiting Time** grows steadily because each subsequent process accumulates more queue delay.
- **Response Time** increases almost linearly since each process starts only after the previous one receives its first quantum.
- **Turnaround Time** rises significantly because it includes both processing time and accumulated wait.

This graph highlights how, in workloads with sequential arrivals and increasing CPU burst lengths, even hybrid schedulers show predictable scaling of delays. It visually reinforces why larger or later-arriving processes experience greater latency despite fair time slicing and priority awareness.