Simulation Analysis of EP, RR, and EP_RR Scheduling Algorithms

SYSC 4001 - Operating Systems

Assignment 3 - Part1 - Simulation Execution

Ibrahim Komeha(101308485) David Mea (101297581)

Introduction:

In this project we have implemented three CPU scheduling algorithms inside a full operating system simulators:

- EP (External Priority, Non-Preemptive): Lower PID = Higher Priority
- RR (Round Robin): Quantum = 100ms
- EP_RR (Priority + RR): Preemptive on priority, but time slice at 100ms

The simulator processes arrival events, CPU bursts, periodic I/O operations, ready/blocked transitions, and memory allocation.

For each simulator we have ran 20 different scenarios which represents:

- Mostly CPU bound workloads
- Mostly I/O bound workloads
- Mixed workloads
- Similar CPU/IO bust lengths
- Priority stress workload

For each simulation we have computed

- Average waiting time
- Average Turnaround Time
- Average Response Time
- Throughput

We have analyzed the results and compared how each scheduling algorithm handled different types of workloads.

Process:

Simulation Input Files:

Created 20 workload scenarios (S01.txt to S20.txt), each containing different combinations of CPU-bound, I/O bounds, and mixed processes.

Schedulars:

- EP: Non-preemptive, selects highest priority (lowest PID).
- RR: Preemptive every 100 ms.
- EP_RR: Combines priority preemption + 100 ms quantum.

Metrics Collected:

For each simulations we have collected:

- Turnaround Time (TAT) = finish time – arrival time
- Wait Time = time spent in READY
- Response Time = first running time – arrival time
- Throughput = number of processes / time of last termination

These were computed using an external Python script using execution.txt.

Result and Analysis:

Scenario	Schedular	Avg TAT	Avg Wait	Avg Response	Throughput
S01	EP	46.67	21.67	21.67	0.04
S01	RR	46.67	21.67	21.67	0.04
S01	EP_RR	46.67	21.67	21.67	0.04
S02	EP	357.67	176.67	176.67	0.0056
S02	RR	483.33	202.33	83.33	0.0056
S02	EP_RR	483.33	202.33	83.33	0.0056
S03	EP	226	106	136	0.0111
S03	RR	150	63.33	63.33	0.0115
S03	EP_RR	250	130	128	0.0111
S04	EP	300	85	180	0.0083
S04	RR	190	45	45	0.0069
S04	EP_RR	287.5	85	155	0.0083
S05	EP	303.33	140	140	0.0061
S05	RR	353.33	190	90	0.0063
S05	EP_RR	353.33	190	90	0.0063
S06	EP	215	90	15	0.0125
S06	RR	215	90	15	0.0125
S06	EP_RR	215	90	15	0.0125
S07	EP	333.33	190	33.33	0.0083
S07	RR	333.33	190	33.33	0.0083
S07	EP_RR	333.33	190	33.33	0.0083
S08	EP	541.67	263.33	25	0.005
S08	RR	541.67	263.33	25	0.005
S08	EP_RR	541.67	263.33	25	0.005
S09	EP	366.25	213.75	23.75	0.0095
S09	RR	366.25	213.75	23.75	0.0095
S09	EP_RR	366.25	213.75	23.75	0.0095
S10	EP	355	215	15	0.01
S10	RR	355	215	15	0.01
S10	EP_RR	355	215	15	0.01
S11	EP	378.75	172.5	172.5	0.0068
S11	RR	343.33	160	68.33	0.0056
S11	EP_RR	383.75	176.25	128.75	0.0069
S12	EP	500	258.33	96.67	0.0042

S12	RR	590	348.33	46.67	0.0044
S12	EP_RR	590	348.33	46.67	0.0044
S13	EP	402.5	113.75	195	0.007
S13	RR	332.5	127.5	47.5	0.0051
S13	EP_RR	425	106.25	195	0.0068
S14	EP	471	339	195	0.0093
S14	RR	495	363	75	0.0093
S14	EP_RR	495	363	75	0.0093
S15	EP	350	185	135	0.0073
S15	RR	357.5	192.5	60	0.0073
S15	EP_RR	357.5	192.5	60	0.0073
S16	EP	435	285	45	0.0083
S16	RR	435	285	45	0.0083
S16	EP_RR	435	285	45	0.0083
S17	EP	300	122.5	75	0.0095
S17	RR	270	150	10	0.01
S17	EP_RR	300	122.5	75	0.0095
S18	EP	500	166.67	250	0.004
S18	RR	475	225	50	0.004
S18	EP_RR	566.67	166.67	200	0.004
S19	EP	475	225	50	0.004
S19	RR	475	225	50	0.004
S19	EP_RR	475	225	50	0.004
S20	EP	738.75	483.75	280	0.0043
S20	RR	888.75	633.75	92.5	0.0043
S20	EP_RR	888.75	633.75	92.5	0.0043

Observation:

CPU-Bound Workloads (S01-S05)
These workloads contain long CPU bursts with little I/O.

- RR consistently lowers both waiting time and response time compared to EP.
- EP sometimes performed well, but became worse when the job lengths differed, since RR prevents long jobs from hogging the CPU.
- EP_RR often matched RR in these scenarios, since quantum preemption beats the scheduling.

RR provides the best performance due to the fair time slicing using the quantum. EP does suffer when long jobs arrive early. EP_RR behaves in the middle of the two but does behave closer to RR.

I/O Bound workloads (S06 - S10)
These workloads generate frequent I/O operations and transition into READY regularly.

- S06-S10 all produced identical responses across EP,RR, and EP_RR
- This happened because frequent I/O interrupts the processes, making preemption and priority less impactful.

All schedulers behave almost identically, I/O frequency, not the CPU scheduling determines the timing.

Mixed workloads (S11 - S15)
These scenarios combine CPU bound and I/O bound processes

- EP had significantly higher response time in mixed workloads
- RR consistently improved response time and usually improved wait time, especially when short or I/O bounded jobs compete with the long CPU bound ones
- EP_RR was able to avoid the starvation that occurred in EP, while still maintaining priority and fairness better than RR

RR gives the best responsiveness. EP_RR was able to balance priority and fairness. EP can delay processes when small PID tasks re-enter READY state frequently.

Similar CPU/IO Bursts (S16-S18)
These workloads contain processes with very similar CPU and I/O behaviour.
For the 3 tests:

- S16: All metrics identical
- S17: RR gave the lowest response time, while EP and EP_RR had the same response time
- S18: EP_RR had higher TAT due to the preemption within the long jobs due to the preemption within long jobs

When job sizes are similar, scheduling decisions matter less. Differences show when priority or quantum affects the ordering as seen in S17.

Priority Stress (S19-S20)
These scenarios were made to test how the schedulers behave when a small PID job and larges PID jobs compete

For the 2 tests:
S19: All metrics identical

S20: metrics were not identical: EP favoured small PID tasks, giving them much better response time, but not fairness.
RR and EP_RR gave better performance but even worse fairness, the large CPU jobs dominate the finish time.

EP is best for finishing jobs fastest on average, but performs worse on responsiveness.
RR is the best for systems with many short jobs.
EP_RR is the best general purpose scheduler, combining priority fairness alongside quantum time slicing.

Final Summary Table (Averages Across All 20 Scenarios)

Schedular	Avg TAT	Avg Wait	Avg Response	Avg Throughput
EP	379.8ms	192.65ms	113.03ms	0.009
RR	384.83ms	210.22ms	48.02ms	0.008
EP_RR	407.44ms	211.02ms	77.90ms	0.009

Turnaround time (TAT)
- EP has the lowest average TAT
- EP_RR is the highest since priority + preemption causes more job switching.

Waiting time:
- EP has the lowest waiting time
- RR and EP_RR are higher and similar since they have more context switching and preemptions.

Response time:
- RR is the quickest
- EP is much slower
- EP_RR is in between

Throughput:
- All three are nearly identical
- The overall work done per unit time is nearly the same across all three schedulers

Across all 20 simulation scenarios, EP had the lowest average turnaround time and waiting time. RR had the best response time, which is in line with the behaviour of time sliced schedulers that allow short jobs to run quickly. EP_RR had the highest turnaround time and wait times due to the more strict / aggressive preemption, but its response time was significantly better than EP's which provides a balanced compromise between response and priority. Throughput was nearly identical across all three schedulers.

Bonus Mark:

While running the 20 simulation scenarios, the simulator produced 'memory.txt' files that showed exactly when each memory partition was used and when it was freed. After reviewing the logs from every scenario, I began to recognize some patterns about how memory was being used depending on the worldload and the scheduling algorithm.

CPU-Bound Workloads (S01–S05)

- Large CPU heavy processes grabbed one of the big partitions early on and stayed there for most of the simulation.
- Since they didn't preform I/O they spent almost the entire simulation in READY or RUNNING which meant the memory that they were using stayed occupied for a very long time.

The memory usage stayed stable, almost nothing was freed until the end, the scheduler didn't matter too much, CPU demand dominated the memory behaviour.

I/O-Bound Workloads (S06–S10)

- Even though these jobs hopped between READY and WAITING, they didn't free their memory during the I/O
- Their memory partitions stayed assigned from the start until termination

Even with a lot of context switching, memory turnover was very low. All three schedulers behaved similarly because memory is only released once a process fully finishes.

Mixed Workloads (S11–S15)

- Large CPU heavy tasks quickly filled the biggest partitions and held onto them for a long time.
- Smaller I/O heavy tasks went in and out much faster so the smaller partitions were seeing more activity.
- In EP, low PID tasks sometimes blocked higher priority partitions for longer, which resulted in delaying other jobs from being admitted.

Mixed workloads created the most interesting and I believe realistic memory patterns. I was able to observe partitions getting freed and reused more often throughout the simulations.

Similar CPU/IO Bursts (S16–S18)

- Memory assignments were more predictable, once a process entered a partition it usually stayed there until the end.
- Turnover was extremely low since all processes were progressing at a similar pace.

Since the jobs behaved the same, all schedulers had nearly identical memory steps / processes.

Priority Stress Tests (S19–S20)

- In EP low PID tasks often took the smaller partitions early and stayed there which would sometimes push bigger, later arriving jobs to wait longer for memory.
- RR and EP_RR did not show this delay as strongly as EP did because they dont follow priority the same way EP has to follow.

Schedular choice did affect memory usage in these tests. EP tended to hold onto partitions longer for high priority jobs while RR and EP_RR resulted in slightly more turnover and more fair memory access.

Observing all the data together, the memory usage has led me to recognize these patterns:

- CPU heavy jobs tend to sit on partitions for quite a long time especially large ones
- I/O bound jobs switch states constantly but still hold memory the entire time.

- EP sometimes kept memory occupied longer than needed.
- RR and EP_RR were a little bit more flexible, which results in faster turnover.
- The most dynamic memory activity happened within the mixed workload.

Overall, tracking memory helped show how scheduling decisions indirectly influence when processes get admitted, as well as how long partitions stay used and which workloads lead to higher or lower rates of memory turnover.