SYSC4001 A3 P1 Report
Repository Link: https://github.com/Xelatrz/SYSC4001_A3_P1

In this report, the results of the outputs from the A3 P1 test cases will be analyzed and compared to see how the different schedulers react with mostly I/O bound, mostly CPU bound processes, and a fairly balanced split of both I/O and CPU burst.

I/O BOUND

Table 1. Metric calculations for student_trace_1 (mostly I/O bound).

Scheduler	Throughput	Avg Wait	Avg Turnaround	Avg Response
EP	0.0169	124.6	174.6	115.6
RR	0.02	173.8	223.8	8.8
EP_RR	0.02	97.8	147.8	97.8

EP:
The EP scheduler had high average wait, turnaround, and response times, as well as a slightly worse throughput compared to the other schedulers. This is because I/O bound processes often return to READY after I/O, but similar to FCFS, EP does not prioritize them. Instead, they get stuck behind long bursts.

RR:
The RR scheduler had the best throughput, tied with EP_RR, and an extremely low average response time. The tradeoff is that the average waiting and turnaround times are significantly higher than the other schedulers. The reason for these results is because RR constantly switches context. I/O bound processes benefit from fast responses, but the few long CPU bursts accumulate many queue re-entries, increasing the wait time, and in turn, the turnaround time.

EP_RR:
The EP_RR scheduler is a mix of the EP and RR schedulers. This scheduler had the best overall balance between all the metrics. It did not have one value that significantly stood out like RR did with average response time, but it has reasonably fast values across the whole table, as well as the lowest average wait time. This is because this scheduler penalizes the large CPU-heavy processes by placing them in the RR scheduler, while I/O-heavy processes get placed in the EP scheduler, which is useful in this mostly I/O bound trace.

CPU BOUND

Table 2. Metric calculations for student_trace_6 (mostly CPU bound).

Scheduler	Throughput	Avg Wait	Avg Turnaround	Avg Response
EP	0.0025	350	750	350
RR	0.0025	616.7	1016.7	83.3
EP_RR	0.0025	350	750	350

EP and EP_RR:
In this mostly CPU bound trace, EP and EP_RR have the exact same metrics which consist of very low average wait and turnaround times compared to RR, but significantly higher average response times. The reason they act this way is because there is no I/O, meaning all processes look the same (CPU bound) and there is no RR demotion or promotion, turning EP_RR essentially into an EP scheduler. The reason for the EP scheduler's metrics is that, when dealing with large CPU bursts, FCFS is typically optimal since there's no context switching, and EP acts similarly to FCFS.

RR:
The RR scheduler has the same throughput and a better average response time, but worse average wait and turnaround times. This is because, as mentioned earlier, when dealing with large CPU bursts, frequently switching context negatively affects the completion times while increasing average response time, which is overall not optimal due to the large negative effects.

MIXED I/O AND CPU BOUND

Table 3. Metric calculations for student_trace_8 (mixed).

Scheduler	Throughput	Avg Wait	Avg Turnaround	Avg Response
EP	0.0035	101	381	96.5
RR	0.0036	252.5	532.5	20
EP_RR	0.0036	101	382.5	20

EP:
The EP scheduler results in relatively low average wait and turnaround times, as well as a relatively equal throughput, but a higher average response time. This is because it depends on the order of the processes. It acts similar to FCFS, so if there’s a large CPU burst, the remaining I/O processes will get stuck without priority.

RR:
The RR scheduler has a similarly fast throughput and average response time to EP_RR, but significantly higher average wait and turnaround times. This is because it quickly serves arriving processes, which is good for response times, but increases total time due to context switches.

EP_RR:
The EP_RR scheduler has relatively fast metrics across the whole table. It does not have one value that stands out compared to rest, but rather shares the fastest time between EP and RR, making it the most efficient. This is because it filters the quick I/O bound processes to be handled like the EP scheduler, while the long CPU bound processes get demoted to an RR scheduler. This is efficient because the short I/O bound processes do not get stuck in the queue behind the large CPU bound processes, and the large CPU bound processes gradually get completed.

CONCLUSION

Overall, RR is typically avoided due to inefficiency, and EP_RR is ideal because it takes the pros of both EP and RR, avoiding the cons wherever possible. In the case of CPU bound workloads, EP is also a valid option because, in this specific case, EP and EP_RR schedulers would act the same. If there’s a choice between these three schedulers in designing a system, EP_RR is the most versatile and efficient.