Report

This is a tabulated summary of the EP scheduler

Metric	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18
Total Processes	1	1	2	2	3	3	3	–	3	3	3	3	4	4	3	5	5	3
Total Time	10	13	20	20	60	37	80	–	5	5	13	3	11	25	31	6	7	60
Throughput	0.10	0.077	0.10	0.10	0.05	0.081	0.038	–	0.60	0.60	0.231	1.000	0.364	0.160	0.097	0.833	0.714	0.050
Avg Wait Time	0.00	0.00	5.00	4.50	13.33	12.33	31.00	–	1.33	1.33	4.33	1.00	5.50	12.00	4.00	2.40	3.20	32.33
Avg Turnaround	10.00	13.00	15.00	16.00	33.33	25.67	58.33	–	3.00	3.00	8.67	2.00	8.25	18.25	14.33	3.60	4.60	52.33
Avg Response Time	0.00	0.00	5.00	1.00	13.33	2.33	23.33	–	1.33	1.33	4.33	1.00	5.50	12.00	4.00	2.20	2.60	20.00

Below is a tabulated summary of the EP + RR scheduler for different scenarios

Scenario	#Proc	CTs	Throughput	Avg TAT	Avg Wait	Avg Resp
1	1	10	0.1	10	0	0
2	1	13	0.077	13	2	0
3	2	10,20	0.1	15	5	5
4	2	18,26	0.0769	22	9	1
5	3	10,30,60	0.05	33.33	13.33	13.33
6	3	26,36,39	0.0769	33.67	23.33	0
7	3	57,66,81	0.037	68	49	0
9	3	1,3,5	0.6	3	0	0
10	3	1,3,5	0.6	3	0	0
11	3	5,8,13	0.231	8.67	3.67	0
12	3	1,2,3	1.0	2	0	0
13	4	5,8,9,11	0.364	8.25	0	0
14	4	10,15,23,25	0.16	18.25	0	0
15	3	46,51,61	0.049	52.67	33.33	0
16	5	1,2,4,5,6	0.833	3.6	2.4	0
17	3	1,11,31	0.097	14.33	4	0
18	5	1,4,5,6,7	0.714	4.6	3.2	0

Below is a tabulated summary of the RR scheduler

Metric	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18
Total Processes	1	1	2	2	3	3	3	–	3	3	3	3	4	4	3	5	3	5
Total Time	10	13	20	20	60	37	80	–	5	5	13	3	11	25	31	6	31	6
Throughput	0.1	0.077	0.1	0.1	0.05	0.081	0.038	–	0.60	0.60	0.231	1.000	0.364	0.160	0.097	0.833	0.714	0.050
Avg Wait Time	0.00	2.00	5.00	6.00	13.33	12.33	17.33	–	1.33	1.33	5.00	1.00	5.50	12.00	17.33	2.00	4.00	1.60
Avg Turnaround	10.00	13.00	15.00	16.00	33.33	25.67	51.67	–	3.00	3.00	9.33	2.00	8.25	18.25	43.33	3.00	14.33	3.40
Avg Response Time	0.00	0.00	5.00	1.00	13.33	2.33	11.67	–	1.33	1.33	5.00	1.00	5.50	12.00	5.00	2.00	4.00	1.40

For each of these tables the scenarios are as follows:

Scenario 1 — Single CPU-bound
EP achieves the highest throughput with zero waiting and response time in this ideal case, while RR slightly worsens turnaround and response time due to time-slice overhead.
Most efficient: **EP**

Scenario 2 — Single IO-bound
EP and RR behave almost identically for a single I/O-bound task, with only minor overhead differences depending on the quantum.
Most efficient: **Roughly equal (tie)**

Scenario 3 — Two CPU-bound
EP keeps waiting and response times low with stable throughput, whereas RR increases both due to sharing the CPU in time slices.
Most efficient: **EP**

Scenario 4 — One CPU + one IO
EP handles the I/O-blocking pattern more effectively, giving lower waiting and turnaround times, while RR adds extra delay and raises response time.
Most efficient: **EP**

Scenario 5 — 3 CPU-bound, staggered arrivals
EP adapts well to staggered arrivals and gives better turnaround times; RR incurs higher waiting due to its quantum and overlapping arrivals.
Most efficient: **EP**

Scenario 6 — 3 IO-bound, frequent IO
RR benefits from quick wakeups and preemption, so frequent I/O is handled efficiently; EP gains

less from short bursts and has slightly more overhead.
Most efficient: **RR**

Scenario 7 — Heavy memory/large processes
EP performance degrades more with very large processes under memory pressure, while RR's fairness helps avoid starvation and can reduce turnaround time.
Most efficient: **RR**

Scenario 9 — IO-bound convoy
RR again reduces the impact of convoys by preventing any single process from monopolizing the CPU, whereas EP tends to favor CPU hogs and worsens the convoy.
Most efficient: **RR**

Scenario 10 — Priority test (Low PID = high priority)
EP respects the priority rule and gives low waiting and turnaround times to high-priority tasks, while RR ignores this priority scheme and performs worse for them.
Most efficient: **EP**

Scenario 11 — RR fairness test
This scenario highlights RR's strength: even sharing of CPU time among tasks and consistent response times, while EP risks starving some processes.
Most efficient: **RR**

Scenario 12 — Mixed workload (2 CPU, 2 IO)
EP tends to minimize turnaround time by finishing bursts efficiently, whereas RR improves responsiveness for interactive/I/O-bound tasks.
Most efficient: **EP for turnaround, RR for response time**

Scenario 13 — Long-running mixed
Over long runs, RR improves overall fairness, while EP may starve longer-running or lower-priority tasks; metrics show a slight edge for RR.
Most efficient: **Slight edge to RR**

Scenario 14 — High memory + IO
I/O-heavy processes get better service under RR, which lowers their waiting times; EP may under-serve them when memory and CPU are both stressed.
Most efficient: **RR**

Scenario 15 — Short bursty processes
EP finishes short CPU bursts quickly with minimal slicing overhead, while RR introduces unnecessary context switches that slow completion.
Most efficient: **EP**

Scenario 16 — Priority inversion test
EP can suffer from priority inversion if not mitigated, whereas RR's regular time-slicing can

reduce inversion effects and improve waiting and response times.
Most efficient: **RR**

Scenario 17 — CPU-bound mix
EP delivers the best throughput and lowest turnaround for a mix of CPU-bound jobs; RR's preemption overhead degrades performance.
Most efficient: **EP**

Scenario 18 — Heavy IO-bound swarm
RR handles many I/O-bound tasks efficiently by frequently rotating among them, giving the highest throughput; EP does not exploit I/O wakeups as effectively.
Most efficient: **RR**