Comparison of Process Scheduling Algorithms

J. Kenneth Wallace

April 20, 2024

This report compares the implementation and performance of four different scheduler algorithms: First-Come First-Serve (FCFS), Shortest Job First (SJF), Round-Robin Scheduling (RRS), and Priority Scheduling (PRI).

1 Implementation

When implementing any of the four, its important to have an underlying data structure capable of storing all process information. For this, I used a static list with an index pointing to the next empty element ("num_elements"). I also implemented a sorting method capable of sorting the "process list" by arrival time, burst time, and priority. This makes it easy for each scheduling algorithm to read through the process list as it will be in the relevant order.

For the FCFS algorithm, order the process list based on arrival time and execute each process in order.

For the SJF and PRI algorithms, you will need to sort by their respective values (SJF == burst_time, PRI == priority). Along with this, you will need to implement a conditional that checks whether or not the current process has arrived based on the current time (which is incremented as each process completes).

Finally, for the RRS algorithm, you need to implement a mixture of all three (FCFS, SJF, and PRI) algorithms along with a unique value stored by each process that contains how much work has been done on it. Each iteration through the list uses a conditional to check whether or not the current process has arrived based on the current time, while also incrementing the wait time of all other processes that have arrived based on the time spent inside the process. This "process time" is calculated by taking the remaining amount of work (burst_time - work_done) and subtracting the round robin quantum time. If it falls below or equal to zero, the process completes and you get the amount of quantum time used and add it to the wait_time of all other processes.

It is safe to say that the Round-Robin implementation is unnecessarily more complex than the other three for no obvious benefit.

2 Performance

All four algorithms use the same, hard-coded processes:

ID	PRIORITY	ARRIVAL TIME	BURST TIME
10	2	0	6
20	4	2	20
30	6	8	10
40	8	6	4
50	0	4	8
60	1	10	13

The results of running the above processes in each algorithm are as follows:

ALGORITHM	TOTAL TURN.	TOTAL WAIT	AVG. TURN.	AVG. WAIT
FCFS	183	122	30.500	20.333
SJF	134	73	22.333	12.167
RRS (2)	247	216	41.167	36.000
RRS (3)	249	219	41.500	36.333
RRS (4)	237	206	39.500	34.333
PRI	182	121	30.333	20.167

Note that RRS (2), (3), and (4) use quantum values of 2, 3, and 4 respectively.

3 Conclusion

Although these tests were done with a small process list, the general pattern because apparent between the four algorithms. Shortest Job First performs immensely better than the other three, while Round-Robin performs considerably worse than the other three. Even if the quantum value of Round-Robin were set higher, it would simply become a First-Come First-Serve algorithm. First-Come First-Serve and Priority algorithms have surprisingly similar results, even though their execution orders are very different.

Overall, in terms of complexity-to-performance, Shortest Job first comes out on top, with First-Come First-Serve a close second simply due to how simple it is to implement. The Priority algorithm, while lacking in terms of performance and simplicity, has some purpose in systems where certain processes need to take precedence over others. While Round-Robin performs horribly compared to the other three, these tests do not account for processes needing to wait for input (Blocked state). The inclusion of this may help Round-Robin in a battle of performance against the other three algorithms. Otherwise, Round-Robin severely lacks in performance and simplicity and should not be used.