Comparative Analysis on Six Classical Scheduling Algorithms

Cyril Kaye Pol
Patricia Nicole Opetina

I. Abstract

Scheduling algorithms play a great role on the overall performance of a machine. This paper focuses on a comparative analysis of the six classical scheduling algorithms with respect to their waiting, turnaround and response time: three of the five scheduling criteria to measure the efficiency of a scheduling algorithm. We first presented established characteristics of the six scheduling algorithms and then provided three test cases to show which of these algorithms is/are exemplary to each of the three scheduling criteria. Test cases included 20, 15 and ten processes with respective arrival, burst and priority, all taken randomly. Graphical presentation of the comparison is likewise shown. Living up to the expectations, Shortest Remaining Time First (SRTF) and Shortest Job First (SJF) both minimized turnaround and waiting time for all of the cases while Round Robin extremely reduced the response time of each process in all the test cases. It is also concluded in this paper that trade-offs do exist in OS.

II. Introduction

Schedules are dime a dozen these days. Just like how individuals needed a schedule to manage his/her tasks, the processor likewise needs a scheduler to '*create*' its schedule to manage work.As a processor is an essential computer resource, CPU scheduling gives a large impact in considering operating system design, implementation and goals.Some of the duties of an operating system are to manage its resources efficiently and allow fast system execution. The OS cannot perform its tasks well without the help of CPU scheduling. Lack of appropriate order of process execution could lead to inconsiderable consequences like inefficient resource use and slow overall performance. To avoid such dilemma, different CPU scheduling algorithms were constructed. Among these algorithms are First Come First Serve (FCFS), Shortest Job First (SJF), Shortest Remaining Time First (SRTF), Non-preemptive Priority, Preemptive Priority and Round Robin (RR). All of these CPU scheduling algorithms were designed to optimize CPU utilization. Now the million dollar question is, which one is better? To answer that question, an experiment was conducted to unveil the most optimal scheduling algorithm given a randomized set of arrival time, burst time and priority. But first, here is a glimpse of the algorithms of interest.

III. Write-up Proper

A. Scheduling Algorithms

1.First Come, First Served (FCFS)

FCFS is considered as the most basic classical scheduling algorithm. As its name suggests, it executes queued processes on the basis of their arrival. It is modeled on real-life customer service and is non-preemptive.

The algorithm for FCFS does not include complicated logic which makes it easier for programmers to implement. It does not suffer from starvation, since every process will basically have its chance to run, maybe sooner (*for processes which entered the system late*), but nevertheless is guaranteed to be executed at some point by the CPU. Drawbacks of FCFS comes from itself being nonpreemptive. Because no preemption of processes occurs, the waiting time of the processes at the end of the queue is largely affected by the execution time of the processes that precedes them. If these preceding processes needs large CPU time, newer processes must have to wait for these long processes to be completed to have a share of the CPU allocated to them. FCFS is best used for batch systems and for environments where processes have short burst times, because late processes will not have to wait long.

2.Shortest Job First (SJF)

SJF prioritizes processes with small CPU bursts. Because it increases throughput as more processes are executed in a shorter time interval and average waiting time is at minimum, SJF is considered provably optimal. Unfortunately, it is difficult to implement as knowing the burst time of a process is impracticable.

3. Shortest Remaining Time First (SRTF)

SRTF algorithm is the preemptive counterpart of SJF. It shares the strengths of SJF such as increasing CPU throughput overtime because of its favor for interactive processes than CPU-intensive tasks. However, just like SJF, it is improbable to implement because of its need for a-priori information of the processes in line for execution. In addition, starvation can happen if longer processes are always preempted by shorter ones, which leads to a longer waiting time of the processes with large bursts.

SJF and SRTF are special kind of priority-based scheduling algorithm, where priority is based on each process' burst time.

4. Nonpreemptive Priority

Priority-based scheduling can either be preemptive or nonpreemptive. A priority is assigned to each process which can be defined either by time, memory requirement or user choice. Its non-preemptive version will simply position a process in the queue where its priority fits, without preempting the currently running process Since processes can have equal priorities, a second scheduling algorithm must be used which adds to scheduling costs.

5. Preemptive Priority

Preemptive priority scheduling preempts a currently running process if it has lower priority than the newly arrived one. Problems such as indefinite blocking can occur since it is always the case that processes with higher priority are executed first. That is, priority-based scheduling can cause lower-priority processes to indefinitely wait for CPU allocation. Priority-based scheduling is suitable when processes is a mix of user-based and kernel processes.

6. Round Robin

Devised primarily for time-sharing systems, Round Robin is the preemptive counterpart of FCFS which optimizes response time. The performance of these algorithm relies heavily on the size of its time quantum. Extremely large time slice would make the policy similar to FCFS. On the other side, too small time slice would result to huge number of context switches.

Round Robin is best used when processes to be scheduled is a mix of long and short processes and that it is the case that tasks will only be executed in completion if all processes have been allocated CPU at a given time.

B. Experimental Setup and Analysis

As stated above, each of the scheduling algorithms has its own advantages and disadvantages, but for the purpose of deducing which standouts in terms of average waiting time, average response time and average turnaround time, we highlight some of the strengths and weaknesses of the algorithms through an experiment.

The experiment consisted of three test cases, with the first having 20 processes scheduled in the ready queue and the second and the third having 15 processes and 10 processes respectively. Belows are the tables illustrating the data used for each of the test cases and its corresponding results.

Arrival Time	Burst Time	Priority
35	1	39
23	12	4
3	43	45
2	2	46
3	49	25
18	38	7
37	41	30
25	4	36
9	3	39
46	35	6
47	28	18
10	30	7
5	49	37
19	41	38
7	9	11
31	23	13
25	44	27
24	44	46
36	33	10
35	45	13

Table 1: *Test Case 1 (20 processes)*

Arrival Time	Burst Time	Priority
41	5	11
19	18	31
14	38	23
7	42	21
40	27	49
48	15	29
22	36	36
37	6	26
44	1	13
39	13	6
48	7	50
45	4	33
40	28	22

23	3	15
33	1	12

Table 2: *Test Case 2 (15 processes)*

Arrival Time	Burst Time	Priority
47	25	27
10	22	49
43	39	18
30	10	43
46	47	47
32	41	3
28	25	28
37	13	47
15	25	49
23	13	9

Table 3: *Test Case 3 (10 processes)*

Scheduling Algorithm	Average Waiting Time	Average Turnaround time	Average Response
FCFS	244.9	273.6	244.9
SRTF	163.4	192.1	150.2
SJF	135.75	159.55	135.75
Nonpreemptive	244.2	272.9	244.2
Preemptive	264.7	293.4	215.9
RR (time slice = 7)	337.2	365.9	36.25

Table 4: *Experiment Results using Test Case 1*

Scheduling algorithm	Average Waiting Time	Average Turnaround time	Average Response
FCFS	121.67	137.93	121.67
SRTF	46.47	62.73	37.6
SJF	55.4	71.67	55.4

Nonpreemptive	83.6	99.87	83.6
Preemptive	77.73	94	76.2
RR (time slice = 7)	89.2	105.47	13.73

Table 5: *Experiment Results using Test Case 2*

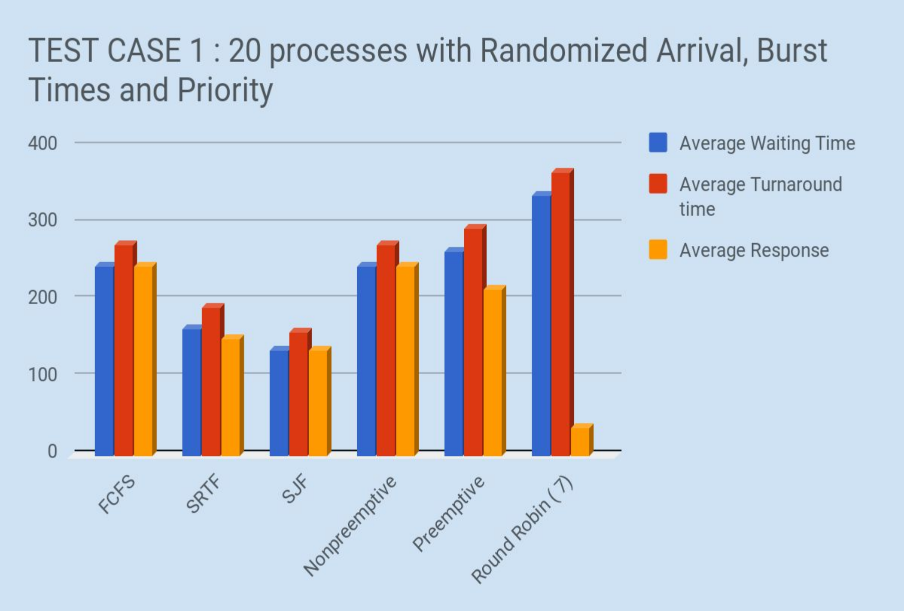
Scheduling Algorithm	Average Waiting Time	Average Turnaround time	Average Response
FCFS	80.6	106.6	80.6
SRTF	65.5	91.5	65.5
SJF	65.5	91.5	65.5
Nonpreemptive	101.8	128.8	101.8
Preemptive	117.8	144.8	91.4
RR(time slice = 7)	141.6	168.6	10.4

Table 3: *Experiment Results using Test Case 3*

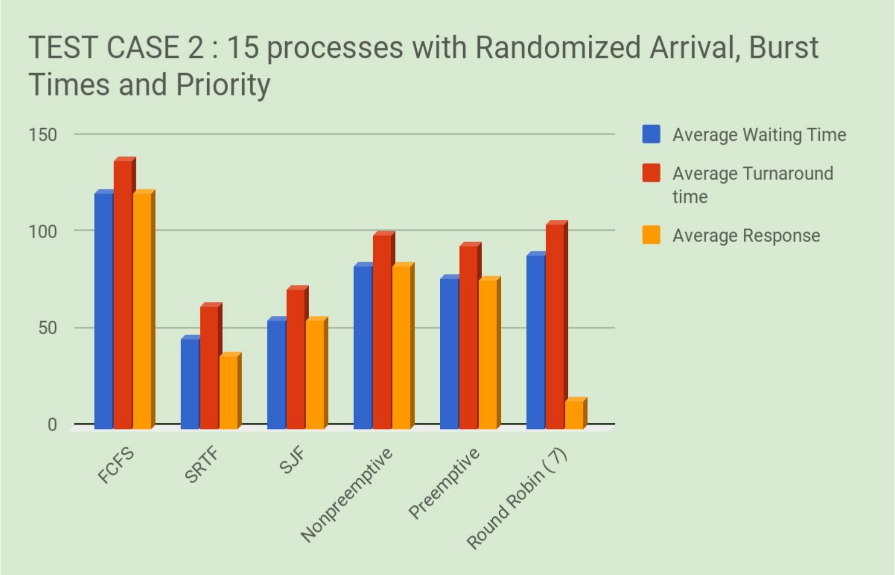
C. Comparison Graph of Test Cases

The bar graph below shows the average waiting, turnaround and response time of all the processes for each scheduling algorithm based on the table above. Both SRTF and SJF reduces the average waiting time and the average turnaround time. Round Robin as shown below greatly reduced the response time.

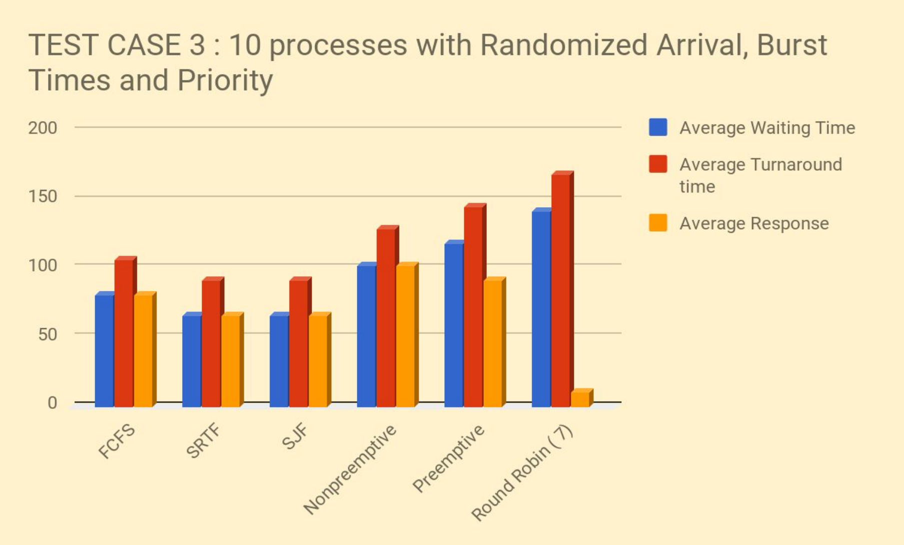
The test cases likewise show that FCFS tends to increase the average waiting, turnaround and response time as the number of processes increases. SRTF and SJF performance is consistent with all of the cases, having the least average waiting and turnaround time. Non-preemptive and preemptive priority scheduling almost has equal performance on all of the test cases. Round Robin extremely reduced the average response time but has increased the average turnaround and waiting time - the largest in all of the test cases. It should also be noted that the time quantum used for Round Robin is a randomly selected number. The choice for the time slice affects the overall performance of the algorithm. It may be degenerated to FCFS if the quantum is too big or may cause too much context switch time if the slice is too small.



TEST CASE 1 : 20 processes with Randomized Arrival, Burst Times and Priority



TEST CASE 2 : 15 processes with Randomized Arrival, Burst Times and Priority



TEST CASE 3 : 10 processes with Randomized Arrival, Burst Times and Priority

IV. Conclusion

In conclusion, when the three aforementioned scheduling criteria are used, SRTF and SJF poses the smallest average waiting and average turnaround time in all test cases. On the other side, RR with quantum 7 holds the best outcome for average response time. However, it must be noted that SRTF and SJF is improbable to implement for it needs a priori information of the processes in line for execution. RR likewise, in all of the test cases, have a large number of waiting time and turnaround time. The choice of for the best scheduling algorithm depends greatly on the specific goals of OS design that outweigh others.

Each scheduling algorithm has its own benefits and drawbacks. Both of its benefits and drawbacks arise from itself being preemptive or not, favoring jobs with smaller burst times or favoring no jobs and allocating all processes a fair share of the CPU. That is, there is always a tool of the trade: one may minimize response time but greatly adds to the overall turnaround and waiting time (Round Robin). One optimizes waiting and turnaround time but may cause indefinite blocking of processes (SJF and SRTF). Some algorithms are mediocre with regards to the CPU scheduling criteria but is useful in special cases, where there is a mix of kernel and user processes (Non-preemptive and Preemptive Priority-Based). One is simple but is usually inefficient(FCFS). With that all said, trade-offs really do exist in OS design.

V. References

"Comparisons of Scheduling Algorithms". Retrieved from https://www.studytonight.com/operating-system/comparision-scheduling-algorithms