ANALYZING CPU SCHEDULING ALGORITHMS ACCORDING TO WAITING TIMES - A CASE STUDY

Ruya SAMLI

Istanbul University, Department of Computer Engineering, Istanbul - Turkey

ruyasamli@istanbul.edu.tr

Abstract: In multiprocessing operating systems, maximum CPU utilization is one of the most important goals. To obtain the maximum rate, the processes in the CPU queue must be scheduled properly. There are several scheduling algorithms to decide this process order as First-Come First-Served Scheduling, Last-Come First-Served Scheduling, Shortest Job Scheduling, Random Scheduling etc. This paper presents a case study to examine which of these scheduling methods is the most efficient one between these policies according to total waiting times.

Keywords: CPU scheduling, waiting time, First-Come First-Served Scheduling, Last-Come First-Served Scheduling, Shortest Job Scheduling, Priority Scheduling, Random Scheduling.

Introduction

In multiprocessing operating systems, the system need a method to decide the order of processes. This process is called CPU scheduling. In the literature there are lots of CPU scheduling policies such as First-Come First-Served Scheduling, Last-Come First-Served Scheduling, Shortest Job Scheduling, Random Scheduling and lots of criteria used for examining the efficiency of the algorithms such as waiting time, response time, turnaround time (Beru, 2015; Carithers and Duncan 2013; Singhoff, 2012; Kohout, 2012; Baskiyar and Meghanathan, 2005). Because most of the examinations about CPU scheduling used total waiting times in the literature, we also use this criteria in this study. The explanations of the methods are given below.

First-Come First-Served Scheduling : The first executed process is the first one in the waiting queue (Kumar et. Al, 2014; Rogiest. et al,2015; Huang, 2014).
Last-Come First-Served Scheduling : The first executed process is the last one in the waiting queue (Harchol-Balter, 2013; Jouini, 2012; Lister, 1993).
Shortest Job First : The first executed process is the shortest one in the waiting queue (Ru and Keung, 2013).
Random Scheduling : The first executed process is a random one in the waiting queue (Tsichritzis and Bernstein, 2012).

In this paper, the CPU scheduling algorithms are analyzed according to total waiting times of the processes. For this reason, 30 process set who has 10 processes and whose burst times are randomly generated (less than or equal to 50ms) are used. The arrival times are accepted as 0. The CPU scheduling algorithms are applied to these process sets and the total waiting times are calculated. This study aims to find the most efficient CPU scheduling method for this case-study. To form more general solutions about the policies, these 30 × 10 sets are chosen as different characteristics. For example in some sets, all of the processes have equal burst times, in some cases, some of them are equal and in the remaining ones, there are no equal burst times in the processes.

Materials and Methods

The CPU scheduling policies can be seperated as preemptive methods and non-preemptive methods. First Come First Served, Last Come First Served, Priority Scheduling, Shortest Job First and Random Scheduling are non-preemptive methods while Round Robin, Shortest Remaining Time are preemptive. In this study the non-preemptive policies are handled because preemptive policies need different types of inputs. The policies that are compared to eachother in this study are First Come First Served, Last Come First Served, Shortest Job First and Random Scheduling. As mentioned above, 10 processes are produced with random burst times in his study. This random burst times are shown in Table 1 which also shows the First Come First Served policy order. Table 1 also shows the processes according to First Come Firs Served Scheduling. Table 2, Table 3, and Table 4 are formed by using the same burst values with Last Come First Served, Shortest Job First and Random Scheduling.

Table 1 : Burst times of processes (FCFS)

Case Number	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
1	1	4	10	8	2	7	13	24	11	4
2	5	5	5	5	5	5	5	5	5	5
3	10	12	5	27	30	30	30	2	2	2
4	4	8	12	16	20	24	28	32	36	40
5	24	1	1	1	12	30	2	7	30	30
6	37	30	33	5	45	30	5	5	24	24
7	37	24	30	10	40	24	30	30	10	10
8	30	10	24	4	15	10	20	24	5	22
9	40	10	10	24	10	10	5	10	10	10
10	24	11	4	14	4	14	12	4	7	30
11	5	5	5	21	18	8	8	20	9	24
12	2	2	2	40	30	10	8	30	10	10
13	2	2	30	33	40	50	8	13	11	7
14	10	2	24	2	7	13	8	5	4	7
15	30	20	10	5	5	5	8	30	5	7
16	17	20	44	30	30	30	8	28	10	14
17	22	22	15	20	24	28	8	14	5	6
18	10	10	10	10	10	10	10	10	10	10
19	12	24	12	24	12	24	12	24	12	24
20	15	13	12	24	49	17	30	2	14	37
21	21	5	15	13	30	34	28	32	33	37
22	22	30	21	5	49	50	2	7	40	30
23	3	28	22	30	30	50	5	5	28	40
24	24	24	3	28	24	24	24	24	24	24
25	17	40	13	34	28	32	10	8	25	9
26	20	41	5	50	2	7	48	12	12	50
27	28	42	30	50	5	5	5	5	5	12
28	13	43	28	50	30	2	2	2	2	17
29	15	20	10	50	28	4	17	17	10	17
30	15	11	22	33	24	24	24	24	50	40

Table 2 : Processes according to Last Come First Served Policy

Case Number	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
1	4	11	24	13	7	2	8	10	4	1
2	5	5	5	5	5	5	5	5	5	5
3	2	2	2	30	30	30	27	5	12	10
4	40	36	32	28	24	20	16	12	8	4
5	30	30	7	2	30	12	1	1	1	24
6	24	24	5	5	30	45	5	33	30	37
7	10	10	30	30	24	40	10	30	24	37
8	22	5	24	20	10	15	4	24	10	30
9	10	10	10	5	10	10	24	10	10	40
10	30	7	4	12	14	4	14	4	11	24
11	24	9	20	8	8	18	21	5	5	5
12	10	10	30	8	10	30	40	2	2	2
13	7	11	13	8	50	40	33	30	2	2
14	7	4	5	8	13	7	2	24	2	10
15	7	5	30	8	5	5	5	10	20	30
16	14	10	28	8	30	30	30	44	20	17
17	6	5	14	8	28	24	20	15	22	22
18	10	10	10	10	10	10	10	10	10	10
19	24	12	24	12	24	12	24	12	24	12
20	37	14	2	30	17	49	24	12	13	15
21	37	33	32	28	34	30	13	15	5	21
22	30	40	7	2	50	49	5	21	30	22
23	40	28	5	5	50	30	30	22	28	3
24	24	24	24	24	24	24	28	3	24	24
25	9	25	8	10	32	28	34	13	40	17
26	50	12	12	48	7	2	50	5	41	20
27	12	5	5	5	5	5	50	30	42	28
28	17	2	2	2	2	30	50	28	43	13
29	17	10	17	17	4	28	50	10	20	15
30	40	50	24	24	24	24	33	22	11	15

As it can be seen from the table, the processes are organized according to their order in the waiting queue, but the first process becomes the last and the last process become the first. This policy is called Last Come First Served scheduling policy.

Table 3 : Processes according to Shortest Job First

Case Number	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
1	1	2	4	4	7	8	10	11	13	24
2	5	5	5	5	5	5	5	5	5	5
3	2	2	2	5	10	12	27	30	30	30
4	4	8	12	16	20	24	28	32	36	40
5	1	1	1	2	7	12	24	30	30	30
6	5	5	5	5	24	24	30	30	37	45
7	10	10	10	24	24	30	30	30	37	40
8	4	5	10	10	15	20	22	24	24	30
9	5	10	10	10	10	10	10	10	24	40
10	4	4	4	4	7	11	12	14	24	30
11	5	5	5	8	8	9	18	20	21	24
12	2	2	2	8	10	10	10	30	30	40
13	2	2	7	8	11	13	30	33	40	50
14	2	2	4	5	7	7	8	10	13	24
15	5	5	5	5	7	8	10	20	30	30
16	8	10	14	17	20	28	30	30	30	44
17	5	6	8	14	15	20	22	22	24	28
18	10	10	10	10	10	10	10	10	10	10
19	12	12	12	12	12	24	24	24	24	24
20	2	12	13	14	15	17	24	30	37	49
21	5	13	15	21	38	30	32	33	34	37
22	2	5	7	21	22	30	30	40	49	50
23	3	5	5	22	28	28	30	30	40	50
24	3	24	24	24	24	24	24	24	24	28
25	8	9	10	13	17	25	28	32	34	40
26	2	5	7	12	12	20	41	48	50	50
27	5	5	5	5	5	12	28	30	42	50
28	2	2	2	2	13	17	28	30	43	50
29	4	10	10	15	17	17	17	20	28	20
30	11	15	22	24	24	24	24	33	40	50

In this table the proceses are ordered according to their burst times in the ascending order. This means the shortest job becomes the first and the longest jobe becomes the last. Finally, another policy called Random Scheduling Policy is applied to the processes. Random Scheduling Policy choses processes randomly as the name implies.

Table 4 : Processes according to Random Scheduling Policy

Case Number	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
1	24	1	10	11	2	7	8	13	4	4
2	5	5	5	5	5	5	5	5	5	5
3	27	2	30	2	30	10	30	12	5	2
4	28	4	40	8	32	20	36	24	16	12
5	24	1	30	1	30	7	30	12	2	1
6	30	5	45	5	30	24	37	24	5	5
7	30	10	40	10	30	24	37	30	24	10
8	22	4	30	5	24	15	24	20	10	10
9	10	5	40	10	10	10	24	10	10	10
10	12	4	30	4	14	7	24	11	4	4
11	18	5	24	5	20	8	21	9	8	5
12	10	2	40	2	30	10	30	10	8	2
13	30	2	50	2	33	11	40	13	8	7
14	8	2	24	2	10	7	13	7	5	4
15	10	5	30	5	20	7	30	8	5	5
16	28	17	14	10	30	20	30	30	8	44
17	22	5	28	6	22	15	24	20	14	8
18	10	10	10	10	10	10	10	10	10	10
19	24	12	24	12	24	12	24	24	12	12
20	24	2	49	12	30	15	37	17	14	13
21	32	5	37	13	30	21	15	33	38	34
22	30	2	50	5	40	22	49	30	21	7
23	30	3	50	5	30	28	40	28	22	5
24	24	3	28	24	24	24	24	24	24	24
25	28	8	40	9	32	17	34	25	13	10
26	41	2	50	5	48	12	50	20	12	7
27	28	5	50	5	12	5	5	30	5	42
28	28	2	50	2	30	13	43	17	2	2
29	17	4	20	10	20	17	28	17	15	10
30	33	24	50	15	11	24	40	24	24	22

Results and Discussion

We constituted computer programs by using MATLAB simulation program and calculated the total waiting times of First Come First Served (FCFS), Last Come First Served (LCFS), Shortest Job First (SJF) and Random Scheduling (RS).

The total waiting times for each process set are calculated according to the algorithms and the results are given below :

Table 5 : Total waiting times

Case Number	FCFS	LCFS	SJF	RS
1	295	461	209	452
2	225	225	225	225
3	749	601	347	790
4	660	1320	660	1024
5	467	775	281	734
6	1228	914	553	1065
7	1251	954	811	1131
8	770	706	499	772
9	789	462	419	662
10	543	573	287	570
11	441	666	359	621
12	572	724	306	706
13	903	861	434	1011
14	411	327	212	404
15	664	461	317	609
16	1161	918	744	934
17	888	588	522	766
18	450	450	450	450
19	780	840	660	852
20	888	1029	601	1016
21	879	1353	886	1048
22	1120	1184	682	1182
23	988	1181	676	1134
24	957	1050	891	940
25	1107	837	659	1043
26	1061	1162	588	1255
27	1173	510	409	832
28	1127	574	380	997
29	916	776	548	686
30	961	1442	911	1272
AVERAGE	814,13	797,46	517,53	839,43

Conclusion

This paper presents a case study to examine which one of the scheduling methods is the most efficient one between First-Come First-Served Scheduling, Last-Come First-Served Scheduling, Shortest Job Scheduling, Random Scheduling according to total waiting times.

As it can be seen easily from the tables, in all the situations Shortest Job First policy is the most efficient one. Because the total waiting time is of course will be the smallest one in all the other policies. Also, when we look at the average times, SJF has the smallest one. The order of the other policies can be changed according to policy but SJF will be the best one in all the cases.

Acknowledgements

The author wishes to thank for the support of the Research Fund of Istanbul University under Project BYP-2016-20712.

References

Baskiyar, S., Meghanathan, N. (2005). *A Survey of Contemporary Real-time Operating Systems*, Informatica 29 (pp. 233–240).

Harchol-Balter, M. (2013). *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*, Cambridge University Press.

http://beru.univ-brest.fr/~singhoff/ENS/USTH/sched.pdf

https://www.cs.rit.edu/~rwd/os1/pdf/0401-Scheduling.pdf

http://www.geekinterview.com/question_details

Huang, K. (2014). *Benchmarking non-first-come-first-served component allocation in an assemble-to-order system*, Annals of Operations Research, 223(1), (pp. 217–237).

Jouini, O. (2012). *Analysis of a last come first served queueing system with customer abandonment*, Computers & Operations Research, 39(12), (pp. 3040–3045).

Kohout, B. (2002). Hardware Support For Real-Time Operating Systems, Master Thesis.

Kumar, M., Panwar, M., Bhargava, S. (2014). *Simple Sequence Oriented Disk (SSOD) Scheduling Algorithm,* International Journal of Computer Applications, 100(2).

Lister, A. (1993). *Fundamentals of Operating Systems,* Springer Science and Business Media, LLC.

Rogiest, W., Laevens, K., Walraevens, J., Bruneel, H. (2015). *When Random-Order-of-Service outperforms First-Come-First-Served*, Operations Research Letters, 43(5), (pp. 504–506).

Ru, J., Keung, J. (2013). *An Empirical Investigation on the Simulation of Priority and Shortest-Job-First Scheduling for Cloud-Based Software Systems*, Proceeding of 22nd Australian Software Engineering Conference, (pp. 78-87).

Tsichritzis, D., C., Bernstein, P., A. (2012), *Operating Systems*, Academic Press.