ANALYSIS OF PROCESS SCHEDULING ALGORITHMS: SJF WITH AND WITHOUT INTERRUPTIONS

[1]JASMINA NOVAKOVIC, [2]SUZANA MARKOVIC, [3]ALEKSANDAR VUKOMANOVIC

[1,2,3] Belgrade business school - Higher education institution for applied studies
E-mail: [1]jasminanovakovic65@gmail.com, [2]suzana.markovic@bbs.edu.rs, [3]avukomanovic2@gmail.com

Abstract - Process scheduling is a key operation for operating systems. The goal of scheduling algorithms is to minimize delay time, response time, and time needed to change the context. SJF (Shortest Job First) is a process scheduling algorithm that selects the one that requires the least execution time in the order of the ready execution process. The SJF algorithm can be with and without interruption. In this paper, the aim of the research is to experimentally analyze the performance of the SJF process scheduling algorithm with and without interruption.

Index Terms - Interrupt, Context Change, Process Scheduling, SJF.

I. INTRODUCTION

Computer users are often not aware of the processes that take place in it. Due to the existence of multiple processes in queues on different devices, it is very important to arrange the processes in the order in which the computer will perform best performance. The problem of scheduling the process dates back to the early stages of computer development. In his article, Rukhsar Khan [1] presented and analyzed basic versions of the FCFS (First Come First Served) and SJF algorithms. Author Andysah Putera Utama Siahaan [2] analyzed process execution in the processor using three basic scheduling algorithms: FCFS, SJF and RR (Round Robin). After studying the work of other authors in this field, we came up with the idea to experimentally analyze the performance of the SJF algorithm with and without interruption using the appropriate data sets. After the introductory consideration, in the second part of this paper we discuss the problem of process scheduling, as well as special system programs - schedulers. The third part discusses the theoretical assumptions of the algorithm SJF scheduling process with and without interruption. In the fourth part, we deal with experimental research and analysis of the results obtained over a specific set of data. In the last part, we made conclusions based on the results of our experiment.

II. PROCESS ALLOCATION

Process allocation aims to allocate processes to the processor for execution, taking into account the response time, bandwidth, and processor efficiency. Deployment policies determine which process, between the ready processes, will be the next in the queue for execution. The process allocation is very important when we have more processes running at any given time and can have a major impact on the utilization and overall system performance [3, 4, 5]. Distributors are special system programs that deploy processes in several different ways. Their main task is to decide which process will next be submitted to the system and that will be executed next. There are several types of schedulers, from which we will mention:

- A long-term scheduler, or receiver, decides which processes are received in the processing system. He selects processes from the order and loads them into the memory for execution;
- Mid-term scheduler - is responsible for interrupting and continuing the process. It mainly makes a replacement, i.e. moves processes from main memory to disk and vice versa;
- Short-term scheduler - decides which process will follow. The second name of this scheduler is "dispatcher".

Appropriate algorithms contain pre-defined steps to solve the problem of scheduling the process. In this text, we have specifically studied the SJF algorithm with and without interruption.

III. SJF ALGORITHM

SJF algorithm selects from the queue job that requires less processing time. With this scheduling algorithm, the process with the shortest expected processing time is chosen to be the next [6]. This means that short process to jump to the head of the queue.

It can be shown that the SJF algorithm is optimal in the sense that for a given set of processes it gives a minimum average waiting time. The problem is that the response time variation increases, especially for long processes, which reduces predictability. One of the problems of using SJF algorithm is that it is necessary to know or estimate the required processing time of each process. In the case of batch processing, this estimate is left to the user as an estimate of the execution time limit, which is then delivered to the operating system. In short-term scheduling, the required processing time of each process can not be known in advance, but can only be predicted. The prediction can be done for example based on pre-history, because the same tasks are often performed, so that statistical data can be collected on

them.

When applying this algorithm, the occurrence of "starvation" for longer processes is possible, as long as there are shorter processes that are constantly reaching. The implementation of this algorithm is not easy, since the execution time of all processes is not known in advance.

The SJF algorithm can be both with and without interruption. With the interrupt algorithm, an estimate of the remaining execution time of an interrupted process is taken into account, in which case the scheduler chooses a process that has the shortest expected remaining processing time. If the new process joins the waiting queue of the ready processes, and if it has a shorter remaining time than the currently running process, the scheduler can interrupt the current process when the new process is ready. The scheduler of algorithm with and without interrupting must have an estimation of the processing time to perform the process selection function, which is why there is a risk of starvation of longer processes.

The interrupted SJF algorithm has better performance in terms of the time required to pass tasks than the non-interrupted algorithm, since short jobs gives a direct advantage over a longer run.

This is a very good policy for processor allocation. However, it is necessary to make some corrections for a job that has been denied a service for a long time. This can be done by introducing some sort of priority that is reviewed whenever an operation is denied access for a longer period of time. An easy way to achieve justice is counting how often the workflow was denied access to the processor. Whenever this number exceeds a certain threshold value, the job must be scheduled for execution in the next time slice.

IV. EXPERIMENTAL RESEARCHE

In an experimental study to analyze the performance of SJF algorithms with and without interruption, we used a set of processes for which we introduced the following assumptions:

-the number of generated processes is 100,
-randomly selected time for executing the given processes.
Random values for time execution of the given processes are generated in Microsoft Excel 2016, using the RANDBETWEEN () function. We also introduced the assumptions for the limit values and time for duration of the process:
- limit values for arrival time are 0 and 25,
- time for duration of the process 0 and 50.
The time period of process monitoring in a processor is expressed in a single time unit.
For the purposes of experimental research, two programs were written in Visual Studio 2017, using the C ++ programming language, to test the operation of SJF algorithms with and without interruption. The results obtained were exported to Microsoft Excel 2016 program in the form of a table for further analysis.

Implementation of the SJF process scheduling algorithm without interruption was done as follows:
1. Sort all processes in a rising order based on the duration of the process.
2. Find the wait time for all processes.
3. Since the first process that comes, it does not have to wait, wait time for the first process will be 0.
4. Calculate wait time for all other processes.
5. Calculate the end time of all processes based on waiting time and processing time.
6. Find the average waiting time of the process.
7. Similarly, find the average end time of the process.
Implementation of the SJF process scheduling algorithm with interruptions was done as follows:

1. Execute until the complete process is completed.
a. Find a process with a minimum residual time in each time range for all the processes that have arrived to the observed moment of time.
b. Decrease the remaining time by 1.
c. Make sure the remaining time becomes 0.
d. Increase the timer to complete the process.
e. Time to complete the current process increase by 1;
f. Calculate the waiting time for each completed process based on the end time, the time of arrival and the duration of the process.
g. Increase the time counter for one.

2. Find the time to complete each process based on the waiting time and the duration of the process.
3. Find the average waiting time of the process.
4. Similarly, find the average time for completing the process.
Fig. 1. shows randomly generated values for processes, waiting times and completion of all interrupted SJF processes. The process table consists of the following columns:
- Process - this column represents the process identifier;
- Duration - indicates the time that the process must spend in the processor;
- Waiting - the moment in which the process was first processed by the processor;
- Completion - indicates the moment when the process is completed.
Waiting represents the time when the process is completed, compared to the time when the process was created and the time of his life. Average waiting is calculated based on waiting for all processes and the total number of processes.
Based on the data from the table and the formula for calculating the average waiting time, we can calculate the average waiting time of the process for the SJF

algorithms with and without interruption. The average waiting time of the process for the SJF without interruption is 850.95, while the average waiting time for the SJF algorithm with interrupt is 838.77.

Process	Duration	Waiting	Ending
1	43	1553	1596
2	32	877	909
3	26	557	583
4	35	1041	1076
5	43	1742	1785
6	26	582	608
7	27	686	713
8	12	101	113
9	14	138	152
10	15	174	189
11	16	206	222
12	38	1313	1351
13	30	796	826
14	37	1197	1234
15	40	1387	1427
16	18	395	413
17	22	432	454
18	4	1	5
19	47	2279	2326
20	48	2425	2473
21	31	862	893
22	32	923	955
23	11	61	72
24	32	944	976
25	36	1090	1126
26	43	1792	1835
27	20	370	390
28	6	3	9
29	45	1956	2001
30	4	7	11
31	10	52	62
32	2	1	3
33	7	36	43
34	19	329	348
35	36	1126	1162
36	47	2337	2384
37	39	1343	1382
38	22	461	483
39	4	7	11
40	2	0	2
41	11	74	85
42	1	0	1
43	46	2101	2147
44	44	1915	1959
45	25	492	517
46	5	23	28
47	17	252	269
48	47	2381	2428
49	43	1841	1884
50	42	1582	1624
51	37	1223	1260
52	20	393	413
53	9	29	38
54	1	0	1
55	12	96	108
56	8	41	49
57	1	1	2
58	16	225	241
59	40	1411	1451
60	26	598	624
61	46	2147	2193
62	41	1495	1536
63	42	1628	1670
64	34	987	1021
65	29	703	732
66	26	623	649
67	17	282	299
68	1	0	1
69	20	424	444
70	4	10	14
71	29	742	771
72	42	1668	1710
73	16	249	265
74	13	123	136
75	46	2206	2252
76	49	2465	2514
77	19	351	370
78	18	316	334
79	14	157	171
80	7	47	54
81	36	1164	1200
82	26	662	688
83	30	821	851
84	14	164	178
85	34	1019	1053
86	29	762	791
87	1	0	1
88	45	2019	2064
89	46	2228	2274
90	12	126	138
91	42	1703	1745
92	45	2058	2103
93	15	192	207
94	25	517	542
95	24	466	490
96	37	1256	1293
97	49	2524	2573
98	40	1472	1512
99	11	73	84
100	43	1869	1912

Fig. 1. SJF with interrupt

The average completion time of the SJF non-interrupt algorithm is 876.81, while the average completion time of the SJF algorithm with interruption is 864.63.
In our research, with randomly generated process values, we have proven that the interrupted SJF algorithm has a better performance than the non-interrupted algorithm, because short work gives a direct advantage over the longer work that is performed. The average waiting time and processing time for the SJF interrupted algorithm is lower than the SJF algorithm without interruption.

We can conclude that the benefits of the SJF algorithm with the interruption are as follows:

1. Short processes are processed very quickly.
2. Also, the system requires very little cost, because it makes a decision only when the process is completed or a new process is added.
3. When a new process is added, the algorithm only needs to compare the process that is executed with the new process, ignoring all the other processes that are currently waiting for execution.

We also noted the following disadvantages for both algorithms:

1. Since the shortest job is first executed, there is a possibility of starvation of the process.
2. Long processes can be kept in the system indefinitely if short processes are continuously added.

CONCLUSION

In our research, we have experimentally concluded that the interrupted SJF algorithm has better performance than the SJF algorithm without interruption. Considering the shortcomings of the existing SJF terminating algorithm, in the next period we will deal with the improvement of algorithms for scheduling the process by attempting to reduce the probability of a starvation process by implementing a counter that counts how often a processor has been denied access to the process. When the value of the counter exceeds the predetermined threshold value, the process will give priority to the execution process.

REFERENCES

[1] R. Khan, „Analysis of Priority Scheduling Algorithm on the Basis of FCFS & SJF for Similar Priority Jobs“, International Journal of Computer Science and Mobile Computing, ISSN 2320–088X, Vol. 4 Issue. 9, September-2015, 2015, pp. 324-331.

[2] A. P. U. Siahaan, „Comparison Analysis of CPU Scheduling: FCFS, SJF and Round Robin“, International Journal of Engineering Development and Research, ISSN: 2321-9939, Volume 4, Issue 3, 2016, pp. 124-131.

[3] B. Đorđević, D. Pleskonjić, N. Maček, Operativni sistemi: teorija, praksa i rešeni zadaci, Beograd: Mikro knjiga, 2005.

[4] W. Stallings, Operativni sistemi: principi unutrašnje organizacije i dizajna, prevod petog izdanja, Beograd: Računarski fakultet CET, 2007.

[5] D. Simić, P. Bataveljić, Organizacija računara i operativni sistemi, Beograd: Fakultet organizacionih nauka, 2011.

[6] D. Milićev, Operativni sistemi, predavanja 2017/2018, Elektrotehnički fakultet, Univerzitet u Beogradu, http://os.etf.rs/, 08.03.2018.

★★★