Article

Milad Scheduling Algorithm (MSA)

Kardan Journal of Engineering and Technology
3 (1) 110–116

Kardan Publications
Kabul, Afghanistan
DOI: 10.31841/KJET.2021.22
https://kardan.edu.af/Research/Kardan_journal_of_engineering_and_technology.aspx#

Abdul Ghaffar
Milad Akbari
Maiwand Yousufzai

Abstract

In today's world, a lot of software is operational in different fields. One of the most important types of software is the operating system. It is an immense set of system software in terms of its functionality and size. It directly interacts with the hardware, fully controls different hardware, and provides a bridge between the user and the hardware. One of the important tasks of the operating system is to schedule the processes so that they can be completed at an acceptable time. Different scheduling algorithms have been proposed, like First Come, First Served (FCFS), Shortest Job First (SJF), and Round Robin Algorithm. In this paper, a new scheduling algorithm has been proposed and its performance has been compared with other existing algorithms, which has proved that Milad's Scheduling Algorithm (MSA) is better in related aspects.

Received: 18 May 21
Revised: 02 Oct 21
Accepted: 28-Nov 21

Keywords: MSA, Algorithm, Software, Operating System

Introduction

Operating System is a type of System Software that has a direct interface with the hardware of the computer system. It provides a communication link between the user and the hardware. Operating Systems are important software in the sense that they carry out multiple tasks and are responsible for the overall operation of the system. One of the important parts of the operating system is the scheduling of the different processes. CPU scheduling is a process in which always one process uses the CPU while the execution of another process is on hold (in waiting for state) due to unavailability of any resource like I/O etc., thereby making full use of CPU. CPU scheduling aims to make the system efficient, fast and fair.

Over the period different scheduling algorithms have been developed and used in operating systems. When it comes to the selection of a scheduling algorithm a criterion needs to be defined explicitly[3][4][5][6][7][8]. Certain criteria have been defined for measuring the quality of scheduling algorithms. Waiting Time and Turnaround time are the important factors in comparing the performance of the algorithms.

Waiting Time is defined to be the time before a process could get the CPU. Turnaround time is the time taken by the process to get completely executed by the CPU. One of the popular scheduling algorithms is Round Robin and is developed specifically for time-shared systems. It is similar to First Come First Served(FCFS) scheduling algorithm whereby a process is

assigned to the CPU until it gets finished. In Round Robin, there is a fixed time quantum whereby a process coming from the queue is given to the CPU according to the quantum time[4-11]. If the process burst time was shorter than the quantum time the scheduler will assign CPU to the next process. If the burst time for the queue was longer than the quantum time then the process will be put at the end of the queue after completing the quantum time. But the disadvantage of Round Robin is that the average waiting time gets large.

2. Literature Review

In the following lines, various scheduling algorithms have been discussed.

Burst Round Robin Algorithm:

In this paper, Helmy and Dekdouk[12] proposed a proportional time-sharing scheduling algorithm according to the burst time of the processes whereby a process having higher burst time will get more burst time and the processes having less burst time will get less burst time. The disadvantage in this approach is that it increases the context switches to more than 50% to the Round Robin.

Changeable Time Quantum Algorithm:

In [4] Samih has proposed that the slice time based on the Round Robin approach should be calculated using integer programming whereby the slice time should neither be large nor short. The disadvantage of this approach is that it doesn't define the limits for the time slice. If the processes are arranged in ascending order then this method behaves like Shortest Job First(SJF) otherwise like FCFS.

Enhanced Round Robin Algorithm:

In [13] Tajwar proposed a dynamic time quantum allocation for each process whereby it is calculated based on the burst time of the processes in the queue. Once the time slice is assigned to a process and it didn't get finished time next time a different time slice is calculated based on the burst times of the remaining processes in the queue. The weak point in this procedure is that the new time slice is calculated based on the average mean of all the burst times which in some cases may be too long for some processes.

An Adjustable Round Robin Algorithm:

In [14] Mostafa and Amano have proposed that the short burst processes should be put first in the queue and then the time quantum should be assigned to each process. The authors propose that this technique will reduce the context switch. But the disadvantage of the approach as we showed in this paper that its average waiting time and average turnaround time gets higher.

In the following lines, experiments have been carried out to show the performance comparison between An adjustable Round Robin algorithm and

Milad's Scheduling Algorithm in terms of Turnaround time and Waiting Time. It has been proved through experiments that MSA performs better.

CPU Scheduling Criteria:

Arrival Time: Time at which the process arrives in the ready queue.

Burst Time: Time required by a process for CPU execution.

Completion Time: Time at which the process completes its execution.

Turnaround Time: Time Difference between completion time and arrival time. TAT = CT – AT **Waiting Time:** Time Difference between turnaround time and burst time. WT = TAT – BT

Milad's Scheduling Algorithm (MSA) :

Milad's Scheduling Algorithm is a preemptive scheduling algorithm in which two queues are linked with stack and hash table. The initial queue is called In_Queue and the queue which takes the remaining processes to the hash table is called Out_Queue. In Milad's Scheduling Algorithm, both stack and hash table contain a program called MTCA (Milad's Time Complexity Analyzer) by which process median average burst time is calculated as variable time quantum.

Both stack and the hash table will store process information in three sections:

a) Process ID

b) Burst time III. TYPES OF SCHEDULING ALGORITHM

There are two types of scheduling algorithms.

Non-preemptive Scheduling Algorithm:

When a process enters the state of running, the state of that process is not changed until it finishes execution or goes to a waiting state.

Preemptive Scheduling Algorithm:

Preemptive scheduling is prioritized. The highest priority process should always be the process that is currently utilized.

c) Priority number

Many times it happens that more than one process gets opened and is being executed when another process gets opened. In this case, the operating system has scheduled the processes so that every process gets a fair share of time to execute the processes. The popular scheduling algorithm that is used in today's operating system is the Round Robin scheduling algorithm[1]. It has bee

shown in today's research paper that Milad's Scheduling Algorithm(MSA) performance is much better than the Round Robin and First Come First Serve(FCFS) algorithm[2] in terms of average turnaround time and average waiting time.

III. MSA PSEUDOCODE

- Insert process to In_Queue

- Push the process to stack
- The initial process priority number is 0 as it arrives in the stack
- MTCA (Milad Time Complexity Analyzer) will define the maximum time quantum a process can execute. It will take all process burst time in both stack and hash table adds them up and divide them by the number of the processes.
- The scheduler checks for which process the MTCA time is sufficient to execute. In other words, those processes that cannot complete their execution due to MTCA time will be ignored.
- If there are more than 1 process that can finish execution by providing them MTCA time, the only process that can be selected is the one with the highest priority number.
- While the process is executing, if a new process comes to the In_Queue, the old process will be preempted and it will be inserted into the Out_Queue.
- Since Out_Queue is connected with the hash table, the old process is shifted into o hash table
- As Milad's Scheduling Algorithm implies, the size of the hash table will be constant 100 so the procremainsmain burst time indicates to which location n of the hash table to be inserted. The formula will be: f(x) = x % 100
- If any process is executed, the priority number of that process will decrease and all the remaining process priority numbers will increase
- If the cess in the hash table is selected for re-execution, it will be again pushed to In_Queue and the procedure will continue until all processes have been executed completely. The formula for selecting the process in hash table is: f(x) = x % 100

Advantages of Milad's Scheduling Algorithm:

- No starvation problem
- No convoy effect problem
- Better response time compared to SJF &
- FCFS
- Less context switch compared to Round Robin algorithm
- More throughput since the selected process will have complete MTCA time
- Less TAT & WT as compared to RR algorithm
- The time for fetching the process from the table will be constant O(1) in most cases due to the usage of the hash table in MSA data structures.
- More overload of processes = more efficient MSA algorithm

Experimental Framework

The experiment consists of several input and output parameters. The input parameters consist of arrival time, burst time and the number of processes. The output parameters consist of average waiting time and an average turnaround time.

(I)

Process	Arrival Time	Burst Time	MSA		FCFS		RR	
			TAT	WT	TAT	WT	TAT	WT
P1	0	10	10	0	10	0	10	0
P2	8	12	21	17	14	10	21	17
P3	10	5	5	10	17	22	10	15
P4	15	2	2	15	14	27	7	20

(II)

Process	Arrival Time	Burst Time	MSA		FCFS		RR	
			TAT	WT	TAT	WT	TAT	WT
P1	0	4	4	0	4	0	4	0
P2	2	7	14	9	9	4	24	19
P3	5	5	7	7	11	11	9	9
P4	6	8	14	12	18	16	23	21
P5	8	9	17	16	25	24	25	24

The above experiment has been taken from one of the research papers in [3]. It has been done to prove MSA algorithm is performing better than the proposed algorithm in the paper for the experiments mentioned there.

(III)

Process	Arrival Time	Burst Time	MSA		FCFS		RR	
			TAT	WT	TAT	WT	TAT	WT
P1	0	3	3	0	3	0	5	2
P2	1	6	14	9	8	3	14	9
P3	4	4	6	6	9	9	9	9
P4	6	2	2	6	9	13	5	9

We can observe from the above experiments that the MSA Algorithm has reduced the turnaround time and the waiting time of the process and increased CPU efficiency. The purposed algorithm is better than simple [RR], [FCFS], and [the optimized RR algorithm] as we observed. MSA has the best capabilities for reducing response time, increasing throughput and avoiding convoy effect problems due to the constraints MSA suggests.

(IV)

Process	Arrival Time	Burst Time	MSA		RR		ADRR	
			TAT	WT	TAT	WT	TAT	WT
P1	0	14	60	46	54	40	60	46
P2	0	13	53	40	57	44	23	10
P3	0	12	33	21	59	47	35	23
P4	0	10	10	0	40	30	45	35
P5	0	11	21	10	60	49	56	45

Milad Scheduling Algorithm proved its performance in comparison with [14]. Starvation has been extremely reduced due to MSA efficient responsiveness and maximum possible throughput.

(V)

			MSA		RR		IRRVQ	
Process	Arrival Time	Burst Time	TAT	WT	TAT	WT	TAT	WT
P1	0	15	25	10	61	46	55	40
P2	0	32	103	71	103	71	103	71
P3	0	10	10	0	46	36	10	0
P4	0	26	87	61	101	75	91	65
P5	0	20	45	25	93	73	75	55

As demonstrated in the chart above, the suggested algorithm has shown an extensive capability for both reducing TAT and WT.

Processes will receive CPU response faster as compared to [15].

5. Conclusion

From the experiments above it is proved that the average turnaround time and average waiting time of MSA is better than the FCFS and Round Robin algorithm. This scheduling method has been compared with one of the latest methods [14] and in comparisons above it demonstrates the better performance of MSA in terms of waiting times and turnaround times. This approach can be practically implemented to further check its proper relevance to the scheduling of the processes.MSA has also been compared in the above lines with [15] and has shown through experiments and graphs that it performs better in terms of Turnaround Time and Waiting Time. MSA through comparisons with RR, FCFS and other updated scheduling algorithms have demonstrated that It could be a very promising algorithm to be implemented in today's various operating systems.

Reference

[1] Silberschatz, Galvin and Gagne, Operating systems concepts, 8th edition, Wiley, 2009.

[2] A.S. Tanenbaun, Modern Operating Systems.3rd Edn, Prentice Hall, ISBN:13: 9780136006633, pp: 1104, 2008

[3] A. Singh, P. Goyal and S. Batra. An Optimized Round Robin Scheduling Algorithm for CPU Scheduling. In *International Journal on Computer Science and Engineering,*Vol.02, No. 07, 2010,2383-2385

[4] S. M. Mostafa and S. Kusakabe, "Effect of Thread Weight Readjustment Scheduler on Scheduling Criteria," Inf. Eng. Express, Jan. 2015.

[5] S. M. Mostafa and S. Kusakabe, "Towards Maximizing Throughput for Multithreaded Processes in Linux," Int. J. New Comput. Archit. their Appl., vol. 4, no. 4, pp. 70–78, 2014.

[6] A. Singh, P. Goyal, and S. Batra, "An optimized round robin scheduling algorithm for CPU scheduling," Int. J. Comput. Sci. Eng., vol. 02, no. 07, pp. 2383–85, 2010.

[7] S. Elmougy, S. Sarhan, and M. Joundy, "A novel hybrid of Shortest job first and round Robin with dynamic variable quantum time task scheduling technique," J. Cloud Comput., vol. 6, no. 1, pp. 0–12, 2017.

[8] S. M. Mostafa, "Proportional Weighted Round Robin: A Proportional Share CPU Scheduler in Time Sharing Systems," Int. J. New Comput. Archit. their Appl., vol. 8, no. 3, pp. 142–47, 2018.

[9] S. M. Mostafa, S. Z. Rida, and S. H. Hamad, "Finding Time Quantum of Round Robin Cpu Scheduling Algorithm in General Computing Systems Using Integer Programming," Int. J. New Comput. Archit. their Appl., vol. 5, no. October, pp. 64–71, Jan. 2010.

[10] A. R. Dash, S. kumar Sahu, and S. K. Samantra, "An Optimized Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum," Int. J. Comput. Sci. Eng. Inf. Technol., vol. 5, no. 1, pp. 07-26, 2015.

[11] M. S. Iraji, "Time Sharing Algorithm with Dynamic Weighted Harmonic Round Robin," J. Asian Sci. Res., vol. 5, no. 3, pp. 131–42, 2016.

[12] T. Helmy and A. Dekdouk, "Burst round robin as a proportional-share scheduling algorithm," Jan. 2007.

[13] M. M. Tajwar, M. N. Pathan, L. Hussaini, and A. Abubakar, "CPU scheduling with a round robin algorithm based on an effective time slice," J. Inf. Process. Syst., vol. 13, no. 4, pp. 941–50, 2017.

[14] S.M.Mostafa,H.Amano," An Adjustable Round Robin Scheduling Algorithm in Interactive Systems",Info. Engg.Express Int.Inst. App.Info. 2019,Vol 5,No.1,11-18

[15] M.K.Mishra,F.Rashid,"An Improved Round Robin CPU Scheduling Algorithm with varying Time Quantum",IJCSEA Vol 4, No.4, 201

About the Authors

Mr. Abdul Ghaffar Department of Computer Science, Faculty of Engineering and Technology Kardan University, Kabul, Afghanistan. <a.ghaffar@kardan.edu.af>

Mr. Milad Akbari, Department of Computer Science, Faculty of Engineering and Technology Kardan University, Kabul, Afghanistan.

Mr. Maiwand Yousufzai, Department of Computer Science, Faculty of Engineering and Technology Kardan University, Kabul, Afghanistan.