International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org
Volume 6, Issue 2, March - April 2017 ISSN 2278-6856

Comparative Study of Various CPU Scheduling Algorithm

Sumit Kumar Nager[1] and Nasib Singh Gill[2]

[1]Department of Computer Science & Applications, MaharshiDayanand University, Rohtak, Haryana, India

[2]Department of Computer Science & Applications, MaharshiDayanand University, Rohtak, Haryana, India

Abstract

Processor scheduling is a mechanism emigrate procedures to diverse states from and to numerous process queues. The number of processes executing together has expanded exponentially from the improvement of multiprogramming and multiprocessor environment. Thus it is mostly desired to schedule the process to attain the best optimal state for the system. Various scheduling algorithms are there to accomplish this task. This paper presents the comparative analysis of various scheduling algorithms such as First Come First Serve, Shortest Job First, Round Robin, Priority, Longest Job First, Multilevel queue, multilevel feedback queue and priority scheduling with their preemptive as well as non-preemptive modes. All these algorithms are presented with their examples and grant charts. These algorithms are judged on the premise of common ready time, average turnaround time, previous understanding that device has to have earlier than scheduling the system and complexity worried in growing those algorithms.

Keywords: Scheduler, LTS, MTS, STS, CPU Scheduling, FCFS, SJF, LJF, Preemptive, Non-Preemptive.

1. INTRODUCTION

The operating system command and synchronises the use of the computer components amid application programs for the numerous users. The operating system acts as the manager of the computer resource (processor, memory, storage, I/O devices) facing a numerous and conflicting request for resources. The operating system must decide how to allocate them to specify programs and users so that it can operate the computer system efficiently and fairly. A single user cannot keep either the CPU of the I\O devices busy at all the time. Multiprogramming increases CPU utilization by organizing job so that the CPU always has one to execute. The operating system keeps several jobs in the memory. It picks and begins to execute one of the jobs in the memory. The jobs may have to wait for some task, such as an I/O operations to complete. In a multiprogramming system, the operating system simply switches to and executes another job when that job needs to wait. The CPU is switched to another job and so on. Eventually the first job finished waiting and gets the CPU back as long as that least one job to execute the CPU never idle. These jobs are defined as the process i.e. any program under execution which requires main memory and processor for executing instructions. This process can be in different states which represent where it is currently residing. The process can have various states but at one-time process will reside only in one state. Initially, the process will in the ***newstate*** i.e. process under creation. Once created, the process will move to the ready state. In the ***ready state***, we have multiple numbers of processes. Select one of the processes from ready state and schedule it on the ***runningstate***. When the process is in the running state it occupies the processor. In running state we have only one process at a time. If the running process completes the execution the process will go to the ***terminalstate***. In case, if the running process requires I/O operation the process will come to ***wait or blockstate***. In the wait state, we can have multiple numbers of processes. Once the I/O operation is completed the process will go to the ready state. It is not going to running state because some other process may present in the running state. As the multiprogramming has increased, the number of processes has also increased. If we have a large number of processes in the ready state then select some process and suspend them and place in the suspended ready state. The suspend ready state in the secondary memory or backing store. When we have sufficient space or memory to manage the process of the ready state, resume the suspended process in the ready state. In a similar manner, multiple numbers of processes are in the wait state. They perform their I/O operations. Several processes can perform the I/O operations in the wait state simultaneously. If we don't have sufficient space to manage the process of wait state we suspend some of the processes and kept in the suspend wait state and when space is available to resume suspended process. When the process completes its i/o operation and is about to move to the ready state but suspend due to the lack of memory, then the process is placed in the suspend ready state.

When process changes from one state to another it's called as context switching which is done by 3 types of schedulers i.e. 1) ***Short Term Scheduler (STS):*** responsible for selecting one of the processes from the ready state for scheduling it on the processor. It is invoked very frequently (milliseconds) (must be fast). 2) ***Long Term Scheduler (LTS):*** responsible for creating and bringing a new process into the system. It is invoked very infrequently (seconds, minutes). Also, it controls the degree of multiprogramming (how many jobs are admitted to run on CPU). 3) ***Medium Term Scheduler (MTS):*** responsible for suspending and resuming the process from the wait state to suspend wait and also in the ready state and suspend ready state.

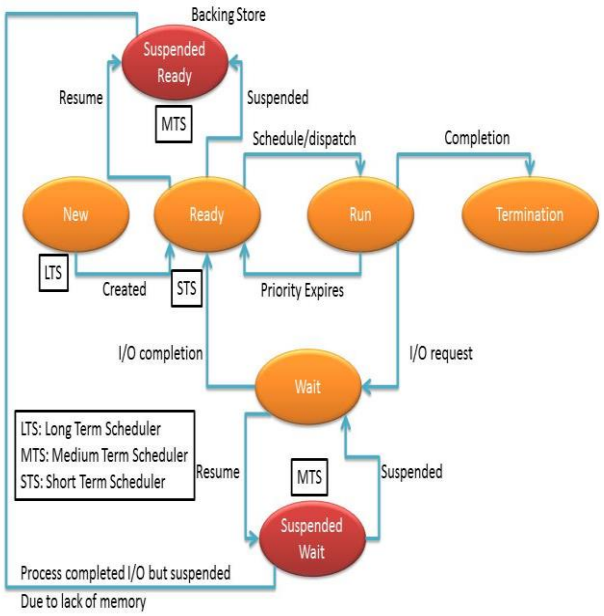

Figure 1 Process Life Cycle and Schedulers

2. WRITINGS EXAMINATION

Many Researchers have introduced various CPU scheduling algorithms from time to time. Some researchers that appropriate with our work are: In the year 2015, Banerjee et al. (2015): proposed another calculation called Optimized Performance Round Robin (OPRR) in which the main concentration on average time quantum which gives result as a less context exchanging and in addition average waiting time and average turnaround time and furthermore lessens the overhead of the CPU by altering the time quantum as per the most astounding burst time of the procedures in the ready queue. [1]. In the year 2014, Adekunle concentrated on the scheduling algorithms utilised for scheduling processes in a multiprogramming system and talked about every algorithm and made a correlation on the premise of eight parameters and different papers that demonstrated no scheduling algorithm perfect fulfilling the conditions and inferred that further reviews which enhance scheduling algorithms need to be done.[2] In the year 2014, Chinmay and Sachdeva investigated the high productive CPU scheduler on the plan of the astounding scheduling algorithms which suits the scheduling objectives and exhibited a state graph delineating the relative investigation of different scheduling algorithms for a solitary CPU that demonstrating which algorithm best for the specific circumstance.[3] In the year 2013, Barman (2013) acquainted another scheduling algorithm with changing time quantum powerfully relying on arrival time and burst time of the procedures based on the experiments and computations. [4]. In the year 2013 Arora proposed CPU scheduling algorithm with enhanced execution utilizing the procedure Pipelining for expanding the accelerate component to enhance its execution by 40-50%[5]. In the year 2011, Behera recommended two processor based CPU scheduling (TPBCS) calculation, where one processor is only for CPU-intensive procedures and the alternate process is solely for I/O-escalated processes that dispatched the procedures to the appropriate processor as indicated by their rate of CPU or I/O necessity to perform better result experimental analysis. [6]

The performance of these schedulers depends on the design and quality of CPU scheduling algorithm. The can be judged on basis of the following criteria: ***CPU utilization***– keep the CPU as busy as possible, **Throughput** – number of processes that complete their execution per time unit,***Turnaround time*** **–** amount of time to execute a particular process (finishing time – arrival time), ***Waiting time***– amount of time a process has been waiting in the ready queue**,** ***Response time*** – amount of time it takes from when a request was submitted until the first response isproduced, not output (for time-sharing environment). ***Priority:*** give preferential treatment to processes with higher priorities. ***Fairness:*** Avoid the process from starvation. All the processes must be given equal opportunity to execute. From these we can deduce the optimization criteria would be maximum CPU utilization, throughput and minimum turnaround time, waiting time and response time.

3. SCHEDULING ALGORITHMS

CPU scheduling is a mechanism to migrate processes to various states from/to various queues. There are numerous CPU scheduling algorithms. In this section, we describe few of them.

3.1. First Come First Serve (FCFS)

Whichever process comes first is scheduled first. The process to be scheduled first is decided by the arrival time of the processor.

Table 1: First Come First Serve

Process Number	Arrival time	Burst Time	Completion time	Turn Around Time	Waiting Time
1	0	4	4	4	0
2	1	3	7	6	3
3	2	1	8	6	5
4	3	2	10	7	5
5	4	5	15	11	6
Average				**34/5=6.8**	**19/5=3.8**

Here ***Completion time*** is the time taken by the process to complete its execution. ***Turn Around Time*** is the time difference between the completion time and arrival time. ***Waiting time*** is the difference between the turnaround time and the burst time. i.e. **TAT=CT-AT** and **WT=TAT-BT.**

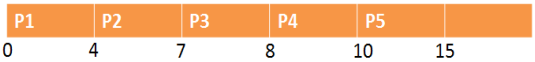

Figure 2 Gantt Chart of First Come First Serve

3.2. Shortest Job First (SJF)

Whichever process has the shortest burst time is scheduled first. It works in two modes ***non-preemptive*** and ***preemptive.***

3.2.1SJF Non-Preemptive

The process which has the shorter burst time is scheduled first. When the first process finishes its execution the next process with the shortest time is scheduled next.

Table 2 SJF Non-Preemptive

Process Number	Arrival time	Burst Time	Completion time	Turn Around Time	Waiting Time
1	6	1	8	2	1
2	3	3	13	10	7
3	4	6	19	15	9
4	1	5	6	5	0
5	2	2	10	8	6
6	5	1	7	2	1
Average				42/6=7	**24/6=4**

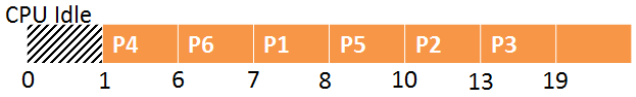

Figure 3 Gantt Chart of SJF Non-Preemptive

3.2.2SJF Preemptive

In preemptive SJF the burst time of process are checked after every unit of time. After completing one unit check process having the shortest burst time will be scheduled next. It's also called as ***Shortest Remaining Time First.***

Table 3 SJF Preemptive

Process Number	Arrival time	Burst Time	Completion time	Turn Around Time	Waiting Time
1	0	7	19	19	12
2	1	5	13	12	7
3	2	3	6	4	1
4	3	1	4	1	0
5	4	2	9	5	3
6	5	1	7	2	1
Average				**43/6=7.1**	**24/6=4**

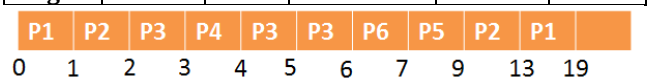

Figure 4 Gantt Chart of SJF Preemptive

3.3. Round Robin Scheduling Algorithm

Whenever the process is going to be scheduled, it's executed for time-quantum units of time.

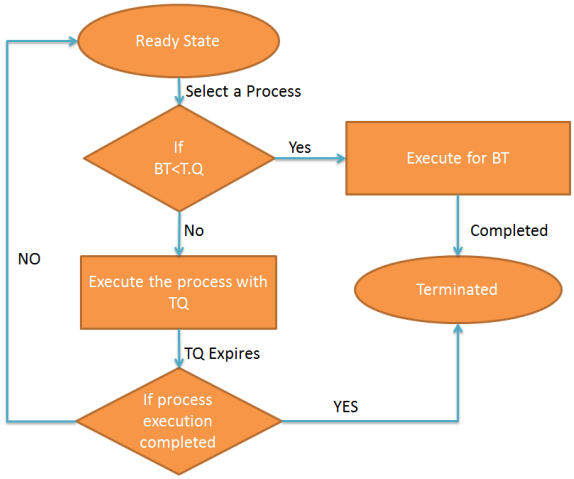

Figure 5 Flow chart of Round Robin Algorithm

Table 4 Round Robin Scheduling

Process Number	Arrival time	Burst Time	Completion time	Turn Around Time	Waiting Time
1	5	5	32	27	22
2	4	6	27	23	17
3	3	7	33	30	23
4	1	9	30	29	20
5	2	2	6	4	2
6	6	3	21	15	12
Average				**128/6=21.3**	**96/6=16**

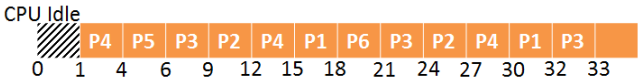

Figure 6 Gantt chart of Round Robin Algorithm

3.4. Longest Job First (LJF)

Whichever process has the longest burst time is scheduled first. It works in two modes ***non-preemptive*** and ***preemptive.***

3.4.3. *LJF Non-Preemptive*

The process which has the longest burst time is scheduled first. When the first process finishes its execution the process with the longest time is scheduled next.

Table 5 LJF Non-Preemptive

Process Number	Arrival time	Burst Time	Completion time	Turn Around Time	Waiting Time
1	3	3	18	15	12
2	1	1	2	1	0
3	2	3	15	3	0
4	4	2	20	16	14
5	6	6	12	9	3
6	2	4	6	7	3
Average				**51/6=8.5**	**32/6=5.3**

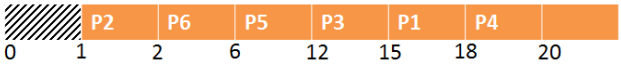

Figure 7 Gantt chart of LJF Non-Preemptive

3.4.4. LJF Preemptive

In preemptive LJF the burst time of process are checked after every unit of time. After completing one unit check process having the longest burst time will be scheduled next. It's also called as ***Longest Remaining Time First.***

Table 6 LJF Longest Time Remaining First

Process Number	Arrival time	Burst Time	Completion time	Turn Around Time	Waiting Time
1	1	2	18	17	15
2	2	4	19	17	13
3	3	6	20	17	11
4	4	8	21	17	9
Average				**68/4=17**	**48/4=12**

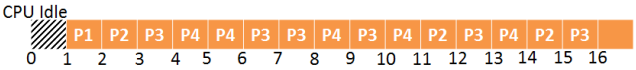

Figure 8 Gantt chart of Longest Time Remaining First

3.5. Priority Scheduling

In this algorithm, a priority is assigned to every process and the process is scheduled on the basis of it. The process that has the highest is executed first then the lower. It works in both preemptive and non-preemptive manner.

Table 7 Priority Scheduling

Priority	Process Number	Arrival time	Burst Time	Completion time	Turn Around Time	Waiting Time
4	**1**	1	4	18	15	12
5	**2**	2	2	14	1	0
7	**3**	2	3	10	3	0
8	**4**	3	5	8	16	14
5	**5**	3	1	15	9	3
6	**6**	4	2	12	7	3
Average					**62/6=10.3**	**47/6=7.8**

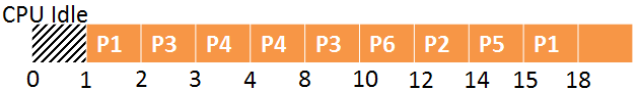

Figure 9 Gantt chart of Priority Scheduling

3.6. Multilevel Queue Scheduling

In this process are partition into different queues and each has its own scheduling algorithm. Depending upon the priority of the process, in which ready queue the process has to be place, is decided. Generally, the high priority process is placed at the top level ready queue and low priority processes are placed in a bottom level ready queue. If this strategy is followed then the process are placed at bottom level ready queue will suffer from starvation.

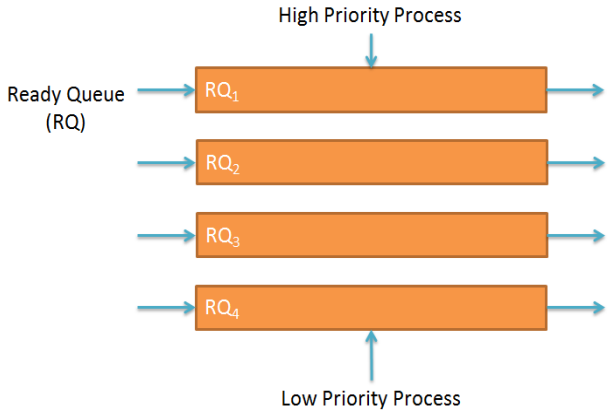

Figure 10 Multilevel Queue Scheduling

3.7. Multilevel Feedback Queue

This scheduling is same as for multilevel queue scheduling but the processes which have no completed its execution at the top level is interrupted and placed in the next level ready queue to remove starvation.

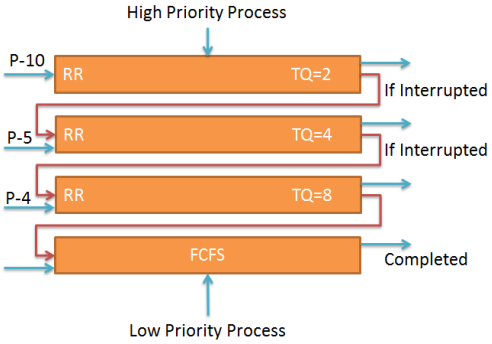

Figure 11 Multilevel Queue Scheduling

4. COMPARATIVE ANALYSIS OF SCHEDULING ALGORITHMS

Following table consist of the analysis of upper mentioned scheduling algorithm on the basis of multiple factors that affect the performance of the computer system.

Table 8 Analysis of scheduling algorithms

S. No.	Algorithm	Implementation Complexity	Preemption	Prior Knowledge of priorities	Starvation	Performance
1	**First Come First Serve**	Easy (FIFO Queue)	No	No	No	large average waiting time
2	**Shortest**	Easy to impleme	No	No	Yes	Minimum

	Job First	nt in batch system				averag e waitin g time
3	**Short est Rema ining Time First**	Impossi ble to implem ent in interacti ve systems	Yes	No	Yes	Short jobs are given prefere nce
4	**Roun d Robi n**	Easy	No	No	No	Fixed time to each proces s
5	**Long est Job First**	Easy	No	No	Yes	Large averag e turnaro und time
6	**Long est Rema ining Time First**	Impossi ble to implem ent in interacti ve or real time systems	Yes	No	Yes	Longer job are given prefere nce
7	**Priori ty Non-Pree mptiv e**	Mildly Comple x	No	Yes	Yes	Best for batch system s
8	**Priori ty Pree mptiv e**	Comple x	Yes	Yes	Yes	Good but proble m of starvat ion
9	**Multi - Level Queu e**	Comple x	No	Yes	Yes	Good but proces s suffer from starvat ion
1 0	**Multi - Level Feed back Queu e**	More Comple x	No	Yes	No	Starvat ion proble m is remov ed

Following table illustrate the advantages and disadvantages of the discussed algorithms.

Table 9 Advantages and Disadvantages of scheduling algorithms

S. No.	Algorithm	Advantages	Disadvantages
1	**First Come First Serve**	Scheduling overhead is minimum	Throughput is low
2	**Shortest Job First**	Minimize waiting time	Prior knowledge of length of next CPU request is required
3	**Shortest Remaining Time First**	Short process executes very fast	Starvation is possible
4	**Round Robin**	Executes all process fairly	Hard to maintain time quantum
5	**Longest Job First**	Very easy to implement	Monopolize CPU
6	**Longest Remaining Time First**	Context switching occurs only after process completion	High turnaround time
7	**Priority Non-Preemptive**	High priority task are done faster	Get complex when priority selection process if not fair
8	**Priority Preemptive**	Waiting time gradually increases for equal priority process	Scheduling overhead is neither minimal nor significant
9	**Multi-Level Queue**	Different scheduling algorithm for different kind of process	Problem of starvation
10	**Multi-Level Feedback Queue**	No problem of starvation	Extra context switching is required

5. CONCLUSION AND FUTURE SCOPE

From the comparative study, it's been deliberated that scheduling is one of the utmost significant responsibilities of the operating system. Depending upon the type and requirements to make the most appropriate system a good scheduling algorithm must be chosen. All the necessary factors i.e. waiting time, response time, turnaround time must be considered to make that decision. First come first serve is the simplest scheduling algorithm and most easy to implement. It is most suitable for interactive and batch systems. The shortest job first algorithm works in both preemptive and non-preemptive manner. It is also easy to implement but the length of next CPU request must be known. Round Robin algorithm treat all process equally and allocate the same time quantum to every process but

it's very hard to decide the proper time quantum. It's best suited for time sharing environment. When in some system the process takes too much time of CPU to execute and then it's best to prefer long time job first algorithm. Sometimes when the process has higher priority than other, those are scheduled by priority scheduling algorithm and it's also work in together preemptive and non-preemptive manner. In this algorithm lower priority process suffers from starvation. Later on, all these are scheduling algorithm are combined to make a new scheduling algorithm called multilevel queue scheduling where process with similar priority are put in the same queue and those queues are scheduled by other scheduling algorithms. The multilevel queue scheduling algorithm faced the problem of starvation to remove it multilevel queue is transformed to multilevel feedback queue scheduling algorithm.

As the era of the computer has evolved new computer architectures are brought up which consist of multiple processors working together to accomplish the different task. Now few processors can be selected to perform the task of lower priority whereas other will work for higher priority processes. Their scheduling must be kept adaptive so that they can transfer the load to another processor to make the system optimal.

References

[1] Banerjee, P., Kumari, A. and Jha, P. Comparative performance analysis of optimized performance round robin scheduling algorithm (OPRR) with an based round robin scheduling algorithm using dynamic time quantum in real time system with arrival time. International Journal Computer Science and Engineering (IJCSE), May 2015,Vol. 03, No. 05, pp. 309-316.

[2] Adekunle, Y., A., Ogunwobi, Z., O., Jerry, A., S., Efuwape, B., T., Ebiesuwa, S. and Ainam J., P. A Comparative Study of Scheduling Algorithms for Multiprogramming in Real-Time Systems. International Journal of Innovation and Scientific Research© 2014 Innovative Space of Scientific Research Journals, November 2014, Vol. 12, No. 1, pp. 180-185.

[3] Chinmay, V. and Sachdeva, S. Comparison Study of Processor Scheduling Algorithms. International Journal of Innovative Research in Technology© 2014 IJIRT, 2014, Vol. 1, No. 6, pp. 1408-1412.

[4] Barman, D. Dynamic Time Quantum in Round Robin Algorithm (DTQRR) Depending on Burst and Arrival Time of the Processes. International Journal of Innovative Technology and Exploring Engineering (IJITEE), March 2013,Vol. 2, No.4, pp. 60-64.

[5] Arora, H., Arora, D., Goel, B., Jain, P. An Improved CPU Scheduling Algorithm. International Journal of Applied Information Systems (IJAIS), Foundation of Computer Science FCS, New York, USA, December 2013, Vol. 6, No. 6, pp.7-9.

[6] Behera, H., S., Panda, J., Thakur, D. and Sahoo, S. A New Proposed Two Processor Based CPU Scheduling Algorithm with Varying Time quantum for Real Time Systems, April 2011, Journal of Global Research in Computer Science, Vol. 2, No. 4, pp. 81-87.

AUTHOR

Sumit Kumar Nagerreceived the B.Tech degrees in Computer Science and Engineeringfrom National Institute of Technology, Kurukshetra in 2013.After that worked as a research analyst in Grey B Research and Services Pvt. Ltd. Joined M.tech program 2014-16 in Computer Science and Engineering from MaharshiDayanand University, Rohtak.