International Journal of Scientific Research & Engineering Trends
Volume 7, Issue 4, July-Aug-2021, ISSN (Online): 2395-566X

A Comparative Review of CPU Scheduling Algorithms

Muskaan Pirani, Denish Ranpariya, Mihir Vaishnav
Department of Computer Engineering
Devang Patel Institute of Advance Technology and Research
Charotar University of Science and Technology
18dce097@charusat.edu.in, 18dce102@charusat.edu.in, 18dce136@charusat.edu.in

Abstract- **Multiple programmes can run in memory at the same time, allowing for CPU and I/O overlap. The problem of selecting a process from the ready queue to be performed by the CPU is addressed by CPU scheduling. Because of the need to adjust and evaluate the operating system as well as assess the performance of actual applications, developing a CPU scheduling algorithm and understanding its effect is challenging and time consuming. Since the processor is such a valuable resource, CPU scheduling is critical for achieving the operating system (OS) design goals. The aim of CPU scheduling is to reduce average turnaround time and average waiting time so that as many processes as possible can run at any given time, maximising CPU utilisation. This paper aims to compile a list of the most common CPU scheduling algorithms that have been proposed so far. FCFS, SJF, SRTF, Round Robin, Priority scheduling, HRRN, and LJF are some of the algorithms we look at.**

Keywords- Scheduler; Dispatcher; FCFS; SJF; SRTF; Round Robin; Priority Scheduling; HRRN; LJF.

I. INTRODUCTION

Multiprogrammed operating systems rely on CPU scheduling. The aim of multiprogramming is to get the most out of your CPU by running certain processes all of the time. Only one process can run at a time in a uniprocessor system; all other processes must wait until the CPU is free. Almost all machine resources are programmed until they are used, which is a primary operating system feature. As a result, since the CPU is one of the most important computer resources, it is crucial to the design of an operating system [1].

Long-term schedulers (also known as work schedulers), mid-term or medium-term schedulers, and short-term schedulers (also known as dispatchers or CPU schedulers) are all examples of schedulers used in operating systems.The scheduling and process transition state is depicted in Figure 1.

The major scheduling algorithms are described in this paper, along with their relative advantages and disadvantages. Basic scheduling algorithms such as FCFS, NP-SJF, SRTF, Round Robin, LRTF, and Priority based algorithms are covered first.

1. Long Term Scheduler:

The long-term scheduler chooses which programmes should be admitted to the system for execution and when, as well as which ones should be terminated. The degree of multiprogramming in multitasking systems is controlled by the long term scheduler. It adheres to such policies that determine which task will be chosen if several tasks are submitted or if the system will approve a new task submission.

The trade-off between degree of multiprogramming and throughput is obvious: all processes compete for a reasonable share of CPU time, and the more processes there are, the less time each of them has to execute.

2. Medium Term Scheduler:

The medium-term scheduler decides when a process should be stopped or restarted. Swapping is the task performed by a medium-term scheduler. Since medium term scheduling is mainly concerned with memory management, it is often included in an operating system's memory management subsystem.

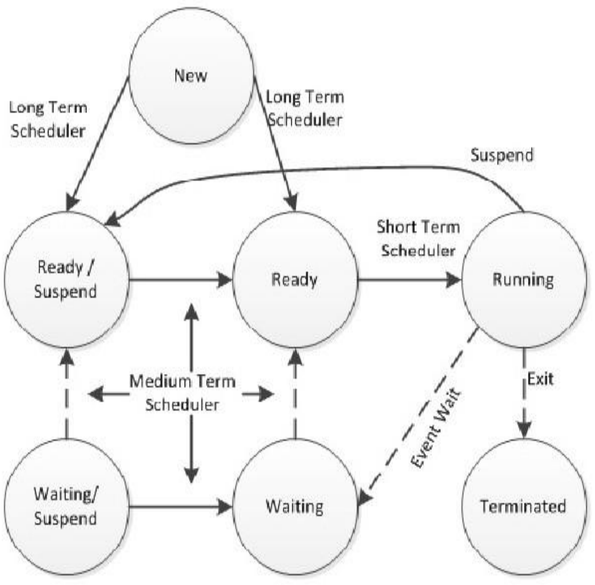


Fig 1. Scheduling and Process State Transition.

3. Short Term Scheduler:

- The short-term scheduler (also known as CPU scheduler) selects a process from the ready-to-run processes in memory to assign to the CPU. Decisions about CPU scheduling can be made when a process:
- Changes from a running to a waiting state (for example, an I/O request).
- Changes the state of the machine from running to ready (e.g., Interrupts occur).
- Changes state from waiting to ready (e.g., I/O completion).
- When a phase comes to an end.

Non-preemptive scheduling falls under 1 and 4; otherwise, it is referred to as preemptive scheduling. Once the CPU has been allocated to a process in non-preemptive scheduling, the process holds the CPU and does not release it until the process is terminated or it moves to a waiting state.

4. Dispatcher:
The dispatcher is in charge of saving one process's context and loading the context of another. The dispatcher is the module that gives the CPU power to the processes chosen by the short term scheduler. The dispatcher is in charge of context switching. The dispatcher should be as fast as possible since it is used every time a process is switched. Dispatch latency [1] is the time it takes for a dispatcher to interrupt one process and resume another.

II. SCHEDULING CRITERIA

Different scheduling algorithms exist, and their success can be assessed using a variety of criteria. Different types of processes which are favoured by different algorithms.

The following are some of the criteria:

- **CPU Utilization:** The amount of time it takes for the CPU to stay as busy as possible.
- Throughput is the number of processes that complete in a given amount of time.
- **Burst Time:** The amount of time it takes for a process to complete.
- **Completion Time:** The amount of time it takes for a method to complete its execution.
- **Turnaround Time:** The amount of time it takes to complete a task. It's defined by:

Completion Time - Arrival Time = Turnaround Time (1)

- **Waiting Time:** The amount of time the procedure spent in the queue. It's defined by:

Turnaround Time - Burst Time = Waiting Time (2)

- **Answer Time:** The time it takes from submitting a request to receiving the first response.

III. OPTIMIZATION CRITERIA

We want to increase CPU utilisation and throughput while reducing turnaround, waiting, and response times. In most cases, we want to optimise the average values, but in certain cases, we want to optimise the minimum and maximum values instead.

The following are some optimization criteria:

- Maximum CPU utilization
- Maximum Throughput
- Minimum Turnaround time
- Minimum Waiting time
- Minimum Response time

IV. SCHEDULING ALGORITHMS

The problem of selecting a process from the ready queue to be performed by the CPU is addressed by CPU scheduling. We'll go through the following CPU scheduling algorithms:

1. First Come First Serve (FCFS):
The simplest scheduling algorithm is first come, first served (FCFS). In this case, the method that demands the CPU first is given priority. The FIFO queue is in charge of implementing the FCFS policy. The FCFS algorithm is non-preemptive, which means that once a CPU has been allocated to a process, the process retains that CPU until it terminates or requests I/O. Because of its non-preemptive existence, the FCFS algorithm is problematic for time-sharing systems, as each process does not get a share of the CPU at regular intervals [1] [3].

2. Shortest Job First (SJF):
SJF (Shortest Job First), also known as SJN (Shortest Job Next) or SPN (Shortest Process Next), is a non-preemptive scheduling policy that selects the waiting process with the shortest execution time to run next [2]. Because of its simplicity, doing the shortest job first is beneficial because it reduces the average amount of time each procedure must wait for its execution to be completed [4]. The downside of SJF is that it starves processes that take a long time to complete if short processes are constantly introduced. Another drawback of SJF is that it needs the total execution time of a job to be known prior to execution, which is not possible [1].

3. Shortest Remaining Time First (SRTF):
Shortest Remaining Time First (SRTF) is a preemptive variant of the Shortest Job First scheduling algorithm, also known as shortest remaining time. The method with the shortest burst time remaining before completion is chosen to be executed in this scheduling algorithm. Any process will continue to run until it has completed its task or until a new process is introduced that needs a shorter burst period.

Shortest remaining time has some benefits. For one, short processes are managed easily if a new process is added; a comparison is made between the leftover burst time of the currently executing process and the new process, ignoring all other processes currently waiting to execute. Second, switching is reduced because it only decides to switch new processes when a new process is added or an existing process completes its execution. It has the same capacity for process starvation as SJF. When process times follow a heavy-tailed distribution, this hazard is reduced to a minimum [5].

The time that remains is the shortest since it necessitates precise estimates of the runtime of all processes waiting to execute, first scheduling is seldom utilised outside of specialised environments.

4. Round Robin (RR):
Round Robin scheduling is a technique for sharing CPU time. Each process is allotted a certain amount of CPU time (a time slice or time quantum), and if it isn't completed by the end of the time slice, it is pushed to the back of the process queue, and the next process in line is allocated CPU time [6][7]. If a process doesn't need the rest of its time slice, it can give it up in a common Round Robin version. This may be because it is awaiting a specific event or because it has been completed. Round Robin (RR) is similar to FCFS, but it includes preemption to move between processes [8][9], as seen in Figure 2.

If the Time Slice/Quantum is too short in Round Robin scheduling, too much process juggling occurs, and the whole process becomes sluggish. If the time limit is exceeded, the machine can become unresponsive, wasting time, and emulating First Come First Served. There are several different versions of the Round Robin algorithm that have better results [10][11][12][13].

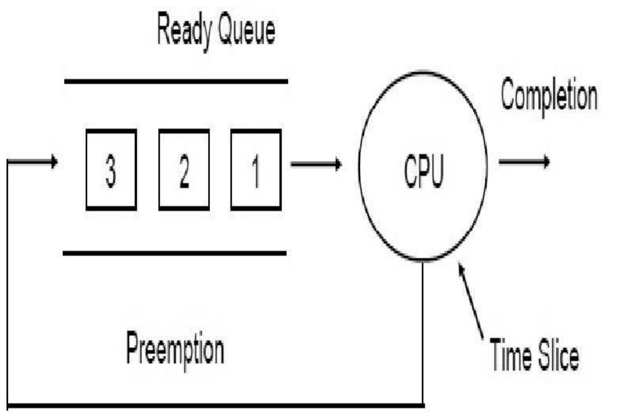


Fig 2. Pictorial representation of Round Robin scheduling.

5. Priority Scheduling:
Each process is assigned a priority number (integer) in Priority Scheduling, and the highest priority process receives the most CPU, as shown in Figure 3. If two or more processes have the same priority, they are scheduled in FCFS order. Preemptive or non-preemptive priority scheduling is possible [14].

A preemptive priority scheduling algorithm will preempt the CPU if the priority of the newly arrived process is higher than the priority of the currently running process.
A non-preemptive priority scheduling algorithm will simply put the new process at the head of the ready queue.

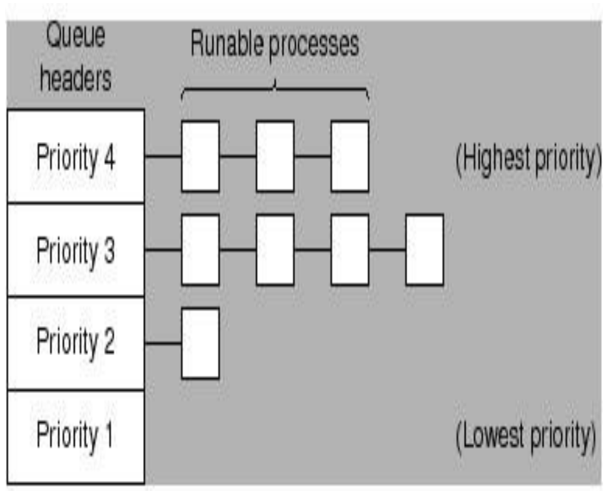


Fig 3. A scheduling algorithm with four priority classes.

A major issue with priority scheduling algorithms is that if a process is ready to run but is waiting for the CPU due to low priority, it is said to be blocked, which is referred to as indefinite blocking or starvation. A steady stream of higher priority processes in a high-processing computer system will prevent a low-priority process from ever receiving the CPU [14].

6. Highest Response Ratio Next (HRRN):
HRRN scheduling [6] is a non-preemptive algorithm, similar to Shortest Work Next (SJN), in which each job's priority is determined by its expected run time as well as the amount of time it has spent waiting. The longer a phase waits, the higher its priority becomes, preventing it from being postponed indefinitely (process starvation).

The following formula is used to determine a process's priority:

$$\text{Highest Response Ratio} = 1 + \frac{\text{Waiting Time}}{\text{Expected Run Time}} \quad (3)$$

Highest Response Ratio (Ratio of Responses) The Next scheduling algorithm was created to address some of the shortcomings of Shortest Job Next (SJN) or Shortest Job First (SJF), such as the difficulty in estimating the runtime.

7. Longest Job First (LJF):
Longest Job First is a scheduling policy that prioritises execution of the waiting process with the longest

execution time. The LJF algorithm is a non-preemptive one. The contradiction behaviour of Longest Job First (LJF) is similar to that of SJF. Although it is believed that processing the shortest job first on the fastest resource will reduce response time, processing the longest job first on the fastest resource will reduce makespan time [15]. However, due to a small increase in response time, LJF will suffer.

V. COMPARISON OF SCHEDULING POLICIES

Assume we have five cycles, P1 through P5, as shown in table 1. Over a collection of data given in Table I, we compare the results of the discussed algorithms.

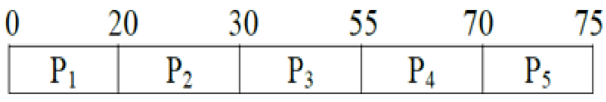
Table 1. Scheduling Dataset.

Process	Arrival Time (Milliseconds)	Burst Time (Milliseconds)	Priority
P1	0	20	1
P2	1	10	2
P3	2	25	2
P4	3	15	4
P5	4	5	3

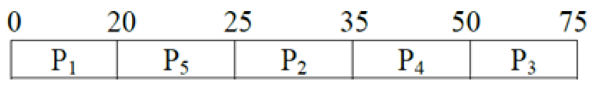
- The jobs have all been completed.
- No new positions will be available before these jobs are completed.
- The amount of time each job will take is already known.
- IO processes are not suspended during the execution of work.

A Gantt chart depicts work processing, from which the average waiting period and average turnaround time are estimated. Table II shows the estimated total turnaround time and average waiting time. Table III shows the assumed modes of operations for comparing all algorithms.

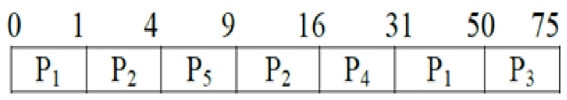
1. First Come First Serve (FCFS):



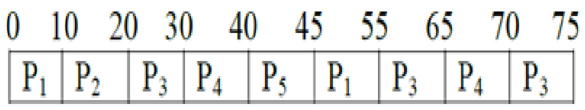
2. Shortest Job First (SJF):



3. Shortest Remaining Time First (SRTF):

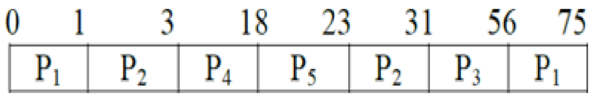


4. Round Robin (Time Slice= 10):

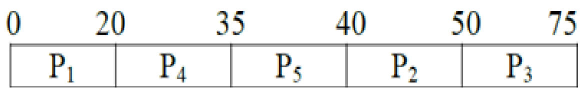


5. Priority Scheduling:
Process P4 with priority 4 is highest while process P1 with priority 1 is lowest.

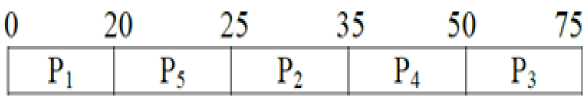
- Preemptive:



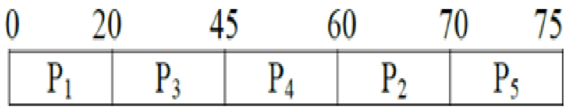
- Non-preemptive:



6. Highest Response Ratio Next (HRRN):



7. Longest Job First (LJF):



VI. SUMMARY OF CPU SCHEDULING ALGORITHMS

We've already gone over a number of different scheduling algorithms. Figure 4 depicts a comparison of the data set's results. IT clearly shows that for the given data collection, Shortest Remaining Time First (SRTF) takes the shortest amount of time to complete processes and has the shortest total waiting time. Longest Job First (LJF) seems to be the most inefficient.

Table III lists the scheduling algorithms, as well as their modes of operation and the conditions that must be understood in order to schedule.

Table 2. Average Turn Around Time and Average Waiting Time.

Algorithm	Average TurnaroundTime (milliseconds)	Average Waiting Time (milliseconds)
FCFS	48	33
SJF	39	24
SRTF	34.6	19.6
Round Robin	51	36
Priority Scheduling (preemptive)	38.6	23.6
Priority Scheduling (non-preemptive)	42	27
HRRN	39	24
LJF	52	37

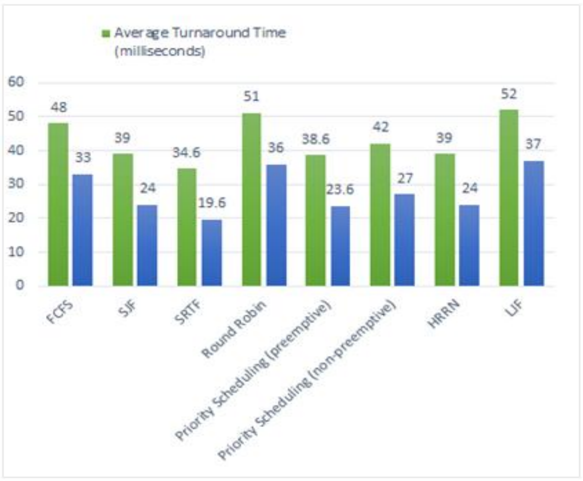


Fig 4. Graph to showing Average Turn Around Time and Average Waiting Time.

Table 3. Summary of Scheduling Algorithms

Algorithm	Mode	Criteria	Starvation
FCFS	Non preemptive	Arrival Time	No
SJF	Non preemptive	Burst Time	Yes
SRTF	Preemptive	Burst Time	Yes
Priority Based	Non preemptive/ preemptive	Priority	Yes
LJF	Non preemptive	Burst Time	Yes
Round Robin	Preemptive	Time quantum/ Time slice	No
HRRN	Non preemptive	Response Ratio	No

REFERENCES

[1] Silberschatz, P. B. Galvin, G. Gagne, Operating Systems Concepts, 7th ed., Wiley publication, 2005. ISBN 0-471-69466-5.

[2] S. Tanenbaum, Modern Operating Systems, 3rd ed., Pearson Education, 2008, ISBN 0-13-600663-9.

[3] J. Patel and A. K. Solanki, CPU Scheduling: A Comparative Study. Proceedings of the 5th National Conference, INDIACom, Computing for Nation Development, March 10-11, 2011.

[4] S. Lupetti, and D. Zagorodnov, “Data popularity and shortest-job-first scheduling of network transfers”, Proceedings of IEEE International Conference on Digital Telecommunications, 2006, pp. 26-26.

[5] M. Harchol-Balter, B. Schroeder, N. Bansal and M. Agrawal, "Size-Based Scheduling to Improve Web Performance", ACM Transactions on Computer Systems, volume 21, issue 2, 2003, pp. 207–233.

[6] W. Stallings, Operating systems: internals and design principles, 4th ed., Prentice-Hall, 2001, ISBN 0-13-031999-6.

[7] R. J. Matarneh, “Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Processes”, American Journal of Applied Sciences, Vol. 6, No. 10, 2009, pp. 1831.

[8] A. Bashir, M. N. Doja and R. Biswas, “Finding Time Quantum of Round Robin CPU Scheduling Algorithm Using Fuzzy Logic, The International Conference on Computer and Electrical Engineering (ICCEE), 2008.

[9] R. Mohanty, H. S. Behera and D. Nayak, “A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis”, International Journal of Computer Applications (0975 – 8887), Vol. 5, No.5, August 2010.

[10] A. Noon, A. Kalakech and S. Kadry, “A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average”, International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 1, May 2011.

[11] R. Mohanty, H. S. Behera, K. Patwari and M. R. Das, “Design and Performance Evaluation of a New Proposed Shortest Remaining Burst Round Robin (SRBRR) Scheduling Algorithm”, In Proceedings of International Symposium on Computer Engineering & Technology (ISCET), Vol. 17, 2010.

[12] 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”, International Journal of Research and Reviews in Applied Sciences (IJRRAS), Vol. 5, Issue 1, 2010.

[13] R. K. Yadav, A. K Mishra, N. Prakash and H. Sharma, “An Improved Round Robin Scheduling Algorithm for CPU scheduling”, International Journal on Computer Science and Engineering, Vol. 02, No.

04, 2010, 1064-1066. Cankaya University, Turkey, http://siber.cankaya.edu.tr/OperatingSystems/ceng328/node1 24.html

[14] [15]Z. R. M. Azmi, K. Abu Bakar, A. H. Abdullah, M. S. Shamsir and W. N. W. Manan, “Performance Comparison of Priority Rule Scheduling Algorithms Using Different Inter Arrival Time Jobs in Grid Environment”, International Journal of Grid and Distributed Computing, Vol. 4, No. 3, September, 2011.