CPU SCHEDULING

Amrish seenu.R
CB.EN.U4CCE22061
Department of Electronics and Communication Engineering
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham, India

Lithika.S
CB.EN.U4CCE22032
Department of Electronics and Communication Engineering
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham, India

Aravind.B
CB.EN.U4CCE2257
Department of Electronics and Communication Engineering
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham, India

Rayhan Muhammed.R
CB.EN.U4CCE22038
Department of Electronics and Communication Engineering
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham, India

Abstract-The fundamental function of an operating system lies in scheduling, where system resources are shared among processes awaiting execution. CPU scheduling, a critical technique, allocates processes to the CPU within specified time intervals. This paper reviews four scheduling algorithms—First Come First Serve (FCFS), Shortest Job First (SJF), Round Robin, and Priority Scheduling—evaluating them based on parameters like running time, burst time, and waiting times. Notably, algorithms may excel in distinct scenarios; FCFS may outperform in short burst times, while Round Robin suits multiple processes consistently. A state diagram is presented for a single CPU, aiding in the comparative study. The objective is to discern the most efficient CPU scheduler, contributing to the design of high-quality scheduling algorithms aligned with specific scheduling goals.

Keywords— Scheduling Algorithms,First Come First Serve(FCFS,)Shortest Job First (SJF),Round Robin,Priority Scheduling,multilevel.

I. INTRODUCTION

In a single-processor system, only one process is capable of execution at any given time, necessitating the queuing of additional processes until the CPU becomes available for rescheduling. The primary goal of multi-programming is to maintain continuous process execution, optimizing CPU utilization [1]. Scheduling, a fundamental function of operating systems, applies to almost all computer resources, with the CPU being a key focus. Consequently, effective CPU scheduling plays a central role in operating system design by determining the sequence in which processes run when multiple runnable processes exist. The impact of CPU scheduling on resource utilization and overall system performance is substantial [2].

Operating systems may incorporate three distinct types of schedulers: a long-term scheduler, alternatively known as an admission scheduler or high-level scheduler; a mid-term or medium-term scheduler; and a short-term scheduler, also referred to as a dispatcher or CPU scheduler. The subsequent sections of this paper are structured as follows:

A. Long-term Scheduler:

The admission scheduler, also known as the long-term scheduler, is responsible for determining which jobs or processes gain entry to the ready queue. Essentially, when an attempt is made to execute a process, the long-term scheduler decides whether to authorize its immediate admission to the currently executing processes or delay it[1]. Consequently, this scheduler holds sway over the selection of processes to run on the system and dictates the level of concurrency to be supported at any given time.

B. Mid-term Scheduler:

The mid-term scheduler engages in the temporary relocation of processes between main memory and secondary memory, such as a disk drive. This process, commonly termed "swapping out" or "swapping in" (and occasionally inaccurately labeled as "paging out" or "paging in"), involves transferring processes to or from secondary memory.

C. Short-term Scheduler:

Also recognized as the CPU scheduler, the short-term scheduler is responsible for deciding which processes in the ready queue, residing in memory, are to be executed (i.e., allocated a CPU) next. This decision occurs following a clock interrupt, an Input-Output (IO) interrupt, an OS call, or another form of signal. Unlike the long-term or mid-term schedulers, the short-term scheduler makes scheduling decisions much more frequently. It can be either preemptive, capable of forcibly removing processes from the CPU, or non-preemptive (voluntary or cooperative), where the scheduler lacks the authority to force processes off the CPU.

The success of a CPU scheduler hinges on the design of a high-quality scheduling algorithm. Such algorithms primarily consider criteria like CPU utilization rate, throughput, turnaround time, waiting time, and response time. The overarching goal of this research is to develop a universally optimal high-quality scheduling algorithm suitable for all job types.

II. LITERATURE SURVEY

A.Introduction:

CPU scheduling algorithms play a pivotal role in optimizing resource utilization and system performance in operating systems. The selection of an appropriate algorithm is crucial as it can significantly impact the execution of various classes of processes. Evaluating and comparing these algorithms based on different criteria is imperative for effective decision-making in choosing the most suitable algorithm for specific computing environments.

B.Criteria for Comparison:

Various criteria have been proposed for evaluating and comparing CPU scheduling algorithms[5]. These criteria encompass essential aspects that influence system performance:

- Utilization/Efficiency: This criterion focuses on maximizing CPU usage by ensuring it remains engaged in useful work as close to 100% of the time as possible.
- Throughput: Maximizing the number of jobs processed per unit of time is crucial to enhance overall system productivity and efficiency.
- Turnaround Time: Minimizing the duration from process submission to completion is critical, particularly for batch users, reducing waiting times for output.
- Waiting Time: The sum of time processes spend in the ready queue needs to be minimized to improve system responsiveness.
- Response Time: For interactive users, minimizing the time from submission to the production of the first response is essential to ensure a smooth user experience.
- Fairness: Ensuring fair CPU resource allocation among processes is crucial for equitable system performance.

C.Role of CPU Scheduler:

The CPU scheduler, activated when the CPU becomes idle, is responsible for selecting processes from the ready queue for execution. This short-term scheduler allocates the CPU to one of the ready processes in memory, thereby determining the sequence of process execution.

D.Literature Synthesis:

In reviewing existing literature, multiple studies have extensively examined and compared CPU scheduling algorithms based on the aforementioned criteria[7]. Researchers have employed various simulation models and real-world experiments to assess algorithm performance.

Numerous algorithms, such as First-Come, First-Served (FCFS), Shortest Job Next (SJN), Round Robin (RR), Priority Scheduling, and Multilevel Feedback Queues, have been evaluated based on their effectiveness in achieving high CPU utilization, maximizing throughput, minimizing turnaround time, reducing waiting and response times, and ensuring fairness in resource allocation.

*E.*Conclusion:

This literature review underscores the significance of evaluating CPU scheduling algorithms based on multiple criteria. The diverse range of criteria ensures a comprehensive assessment of algorithm performance, aiding in the selection of appropriate algorithms for specific computing environments. Future research could delve deeper into comparative analyses using diverse workloads and system configurations, further enhancing our understanding of these algorithms' suitability in various contexts.

III. PROPOSED METHODOLOGY

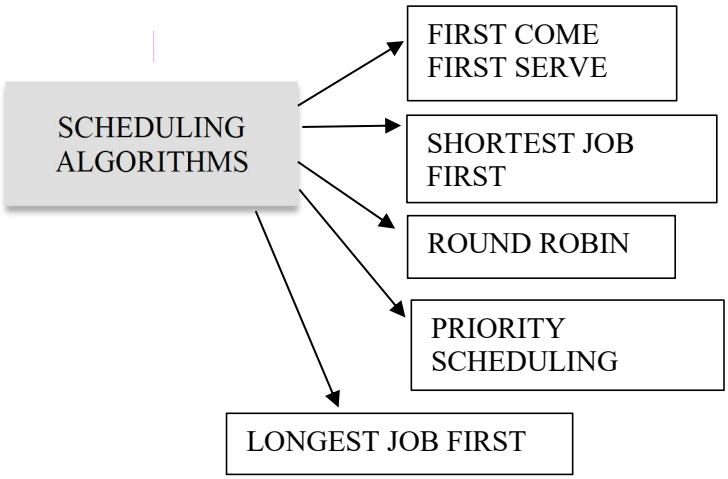

Fig1.Scheduling methods

A.First Come First Serve (FCFS) Scheduling Algorithm:

In the "First Come First Serve" scheduling algorithm, as the name implies, the process that arrives first is granted execution priority. Essentially, the CPU is allocated to the process that makes the initial request for CPU access[4].

•FIFO Queue Arrangement: FCFS operates akin to a FIFO (First In First Out) queue, where the data element added earliest to the queue is the first to be dequeued. The process entering the queue first is the one to exit first.

•Batch Systems Usage: This scheduling approach is commonly employed in batch systems, facilitating a straightforward sequence of process execution.

•Simplicity in Implementation: FCFS is straightforward to understand and implement programmatically. It utilizes a queue arrangement, where a new process enters through the tail of the queue, and the scheduler selects a process from the front of the queue for execution.

•Performance Limitations: Despite its simplicity, FCFS tends to perform poorly as it often results in high average wait times for processes.

•Real-life Example: An illustrative real-life example of FCFS scheduling is the process of purchasing tickets at a ticket counter, where individuals are served in the order they arrive.

Consider the processes P1, P2, P3, P4 given in the below table, arrives for execution in the same order, with Arrival Time 0, and given Burst Time, let's find the average waiting time using the FCFS scheduling algorithm.

Benefits of FCFS:

- Implementation of FCFS is straightforward.
- It follows a first come, first serve approach.

Drawbacks of FCFS:

- FCFS is susceptible to the convoy effect.
- The average waiting time tends to be higher compared to other algorithms.
- Although FCFS is simple to implement, its efficiency is limited.

PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2

The average waiting time =(0+21+24+30)/4=18.75ms

P1	P2	P3	P4	P5
0	21	24	30	32

This is the GANTT chart for the above processes

The average waiting time will be 18.75 ms. For the above-given processes, first P1 will be provided with the CPU resources.Hence,

- The waiting time for P1 will be 0
- P1 requires 21 ms for completion, hence waiting time for P2 will be 21 ms
- Similarly, the waiting time for process P3 will be the execution time of P1 + execution time for P2, which will be (21 + 3) ms = 24 ms.
- For process P4 it will be the sum of execution times of P1, P2, and P3. The GANTT chart above perfectly represents the waiting time for each process.

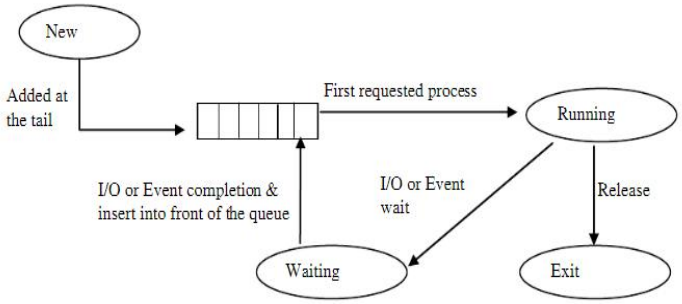

Fig2.Flowchart for First come first serve

B.Priority Scheduling:

- Non-Preemptive Algorithm: Priority scheduling is classified as a non-preemptive algorithm, making it one of the prevalent scheduling methods in batch systems.
- Priority Assignment: In this algorithm, each process is assigned a priority value. The process possessing the highest priority is given precedence in execution.
- Execution Order for Same Priority: When multiple processes share the same priority level, they are executed in a first-come-first-served manner, ensuring fairness.
- Basis for Priority Determination: Priority values are determined based on various factors such as memory requirements, time constraints, or any other resource needs.Consider the below table for processes with their respective CPU burst times and the priorities. Let the arrival time be 0 for all the processes[3].

Advantages of Priority Scheduling:

- Priority Scheduling tends to have a lower average waiting time compared to FCFS.
- It is less complex in its implementation.

Disadvantages of Priority Scheduling:

- One notable drawback of the Preemptive Priority CPU scheduling algorithm is the occurrence of the Starvation Problem. This issue arises when a process has to endure an extended waiting period before being scheduled onto the CPU, a situation referred to as the starvation problem.

PROCESS	BURST TIME	PRIORITY
P1	21	2
P2	3	1
P3	6	4
P4	2	3

The GANTT chart for following processes

P2	P1	P4	P3	P5
0	3	24	26	32

The average waiting time will be=(0+3+24+26)/4=13.25ms

As you can see in the GANTT chart that the processes are given CPU time just based on the priorities.

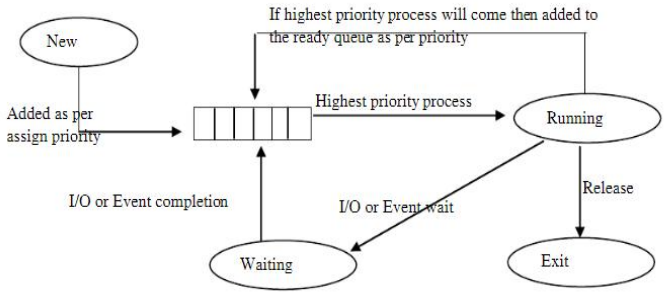

Fig3.Flowchart for priority scheduling

C.Shortest Job First (SJF):

Shortest Job First (SJF) stands as a non-preemptive scheduling policy that prioritizes selecting the waiting process with the shortest execution time for immediate execution. Under this policy, the scheduler ensures each incoming process holds a specific queue position. SJF aims to optimize throughput by favoring shorter execution time jobs.

In this strategy, jobs are arranged in a queue, with the shortest ones at the front and the longest ones at the back. SJF aims to minimize average waiting time by executing smaller jobs before larger ones, thereby improving system efficiency. It essentially prioritizes shorter jobs over longer ones.

One practical application of SJF is in task segmentation, where a portion of a task is designated as mandatory while another part is considered optional. The SJF approach is applied to the optional segment, selecting the process with the shortest execution time for immediate execution[6].

The primary advantage of SJF lies in its significant reduction of average waiting time for different processes. However, challenges such as potential starvation issues and the difficulty in accurately predicting the length of forthcoming CPU requests exist. Despite these limitations, SJF remains an effective scheduling method for minimizing waiting times and prioritizing jobs based on their execution duration.

Advantages of Shortest Job First:

- SJF outperforms the first-come-first-serve scheduling algorithm by reducing the average waiting time.
- SJF is commonly employed for long-term scheduling.

Disadvantages of Shortest Job First:

- A drawback of SJF is the potential occurrence of starvation.
- Predicting the length of future CPU requests can often be challenging and adds complexity to the scheduling process.

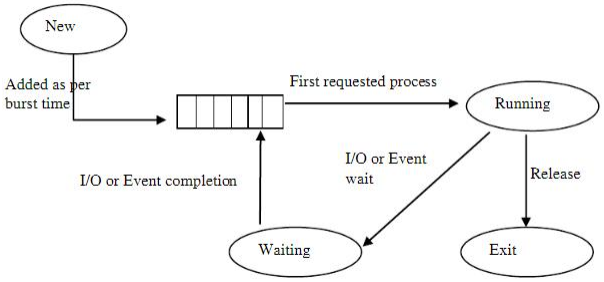

Fig4.Flowchart for shortest job first

D.The Round Robin :

Round robin(RR) scheduling algorithm allocates a specific time slice or quantum to each process, maintaining a queue of ready processes with new jobs added to the queue's tail [43, 44]. Efficient CPU utilization is compromised when the time slice is set too short, while excessively long time slices result in poor response times [7,5]. Although Round Robin is an established and straightforward algorithm, specifically designed for time-sharing systems, its deterministic time allocation nature leads to challenging waiting times and turnaround times.

In Round Robin scheduling, each job is assigned an equal share of the CPU time, promoting fairness among processes due to the time quantum. Despite its advantages over other scheduling algorithms, such as low turnaround and waiting times, challenges arise in meeting deadlines due to elevated waiting times [3]. Introducing a new dynamic quantum scheduling algorithm has been proposed to address these issues. Additionally, recent advancements in Round Robin algorithms aim to decrease context switching and enhance waiting and turnaround times compared to traditional RR scheduling .

Key characteristics of Round Robin:

- Preemptive process scheduling algorithm.
- Each process receives a fixed time, referred to as a quantum, for execution.
- Processes are preempted after their allotted time, and another process executes for a specific duration.
- Context switching is employed to preserve the states of
- preempted processes.

Advantages of Round Robin:

- Round Robin is perceived as fair since each process receives an equitable share of CPU time.
- Newly generated processes are appended to the end of the ready queue.

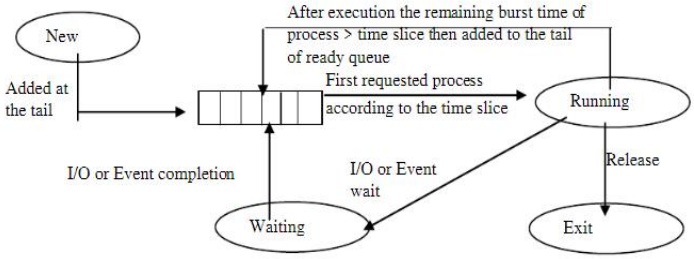

Fig5.Flowchart of Round Robin

PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2

The GANTT chart for following processes

P1	P2	P3	P4	P1	P3	P1	P1	P1	P1
0	5	8	13	15	20	21	26	31	32

The average waiting time will be = 11ms

Consider all the process's arrival time is 0. Here, the time quantum=5

E.Longest Job First (LJF):

In Longest Job First (LJF), the algorithm prioritizes the processing of the job with the longest burst time. This approach is essentially the opposite of Shortest Job First (SJF). LJF does not employ preemptive measures. However, a notable drawback of the LJF algorithm is its tendency to exhibit high

average waiting and turnaround times for a given set of processes. This can lead to a convoy effect, diminishing the overall efficiency of the technique[2,5].

Characteristics of LJF:

- Within a queue of waiting processes, the CPU is consistently assigned to the process possessing the longest burst time.
- When two processes share identical burst times, the tie-breaker involves utilizing FCFS principles, prioritizing the process that arrived first for execution.
- The LJF CPU Scheduling approach can be implemented in both preemptive and non-preemptive variations.

Advantages of LJF:

- No other tasks can be scheduled until the longest job or process completes its execution.
- All jobs or processes typically finish around the same time.

Disadvantages of LJF:

- In general, the LJF algorithm is associated with a notably high average waiting time and average turn-around time for a given set of processes.
- The potential emergence of the convoy effect is a drawback associated with LJF.

IV.RESULT

V.CONCLUSION

The mentioned paper assesses various scheduling algorithms using metrics such as CPU overhead, throughput, turnaround time, and response time. The results indicate that First Come First Serve (FCFS) displays low throughput, low turnaround time, high CPU overhead, and low response time. Shortest Job First (SJF) is characterized by moderate CPU overhead, high throughput, and moderate values for turnaround and response times. Round Robin (RR) is linked to high CPU overhead, medium throughput, medium turnaround time, and high response time. Priority Scheduling is recognized for its medium CPU overhead, low throughput, high turnaround time, and high response time.

REFERENCES

[1] Silberschatz, G. Gagne and P. Galvin, Operating System Concepts, Wiley, pp. 200, 2018.

[2] AL-Bakhrani, A. A. Hagar, A. A. Hamoud and S. Kawathekar, "Comparative analysis of cpu scheduling algorithms: Simulation and its applications", *International Journal of Advanced Science and Technology*, vol. 29, no. 3, pp. 483-494, 2020

[3] H. Parekh and S. Chaudhari, Improved round robin cpu scheduling algorithm: Round robin shortest job first and priority algorithm coupled to increase throughput and decrease waiting time and turnaround time, pp. 184-187, 2016.

[4] R. Kumar Yadav and A. Upadhyay, "A fresh loom for multilevel feedback queue scheduling algorithm", *International Journal of Advances in Engineering Sciences*, vol. 2, pp. 21-23, 2012.

[5] K. Lalit, S. Rajendra and S. Praveen, "Optimized scheduling algorithm", *International Journal of Computer Applications*, pp. 106-109, 2011.

[6] Muhammad Akhtar, Bushra Hamid, Inayat ur-Rehman, Mamoona Humayun, Maryam, Hira Khurshid,An Optimized Shortest Job First Scheduling Algorithm for CPU Scheduling

[7] https://ieeexplore.ieee.org/search/searchresult.jsp?newsearch=true&queryText=cpu%20scheduling