PERFORMANCE EVALUATION OF OPERATING SYSTEMS SCHEDULING ALGORITHMS

Abubakar Sani Jato

Department of Computer science Mai Idris Alooma Polytechnic Geidam, Yobe State. Nigeria.

Key Words:

Scheduling algorithm, CPU Utilization, Round Robin (RR), Shortest Job First (SJF), First Come First Serve (FCFS), Turnaround Time, Waiting Time

ABSTRACT

Scheduling algorithms allow one to decide which threads are given to resource from moment to moment. Various process scheduling algorithms exist and this research will focus on the scheduling algorithms used for scheduling processes in an operating system namely First-Come-First-Served (FCFS), Round Robin (RR), Shortest Job First (SJF) scheduling algorithms. Each algorithm has been discussed and a comparison will be made on the basis of average waiting time which is significant in processes scheduling. In fact, compared to other papers, this research will make use of more types of scheduling algorithms for the analysis. These parameters include CPU utilization, throughput, waiting time, response time. Based on the analysis that will be given after the research, we would recommend the best scheduling algorithm for the design of operating systems which would be more efficient and fast in terms of speed. The objective of the study is to analyse the high efficient CPU scheduling algorithm on design of the high quality and efficient operating system.

INTRODUCTION

An operating system is a collection of software that manages computer hardware resources and provides common services for computer programs. It serves as an intermediary system between users and computer hardware as well as provides users with an environment in which programs can be conveniently and efficiently executed. System Scheduling is the process of deciding how to assign resources to different tasks. These resources may be virtual computation elements such as threads, processes or data flows, which are in turn scheduled onto hardware resources such as processor, network link or expansion cards. A scheduling algorithm, or scheduling policy, is the sequence of steps that the scheduler makes to perform these decisions while the scheduler is what carries out the scheduling activity. Schedulers are often implemented to keep all the computer resources busy (such as load balancing), allow multiple users to share system resources effectively and achieve a target quality of service. Scheduling is fundamental to computing itself. As an intrinsic part of the execution model of a computer system, the concept of scheduling makes it possible to achieve computer multitasking with a single CPU (Abraham, Peter &, Greg, 2013).

A scheduler may aim at one of many goals such as maximizing throughputs, minimizing response time or minimizing latency, maximizing fairness. In practice these goals often conflict (e.g. throughputs vs. latency), thus a scheduler will implement suitable compromise. Preference is given to systems concerned depending on the user needs and objectives.

The analysis of the algorithm is determined by the number of resources (such as time and storage) necessary to execute them. Most algorithms are designed to function with inputs of arbitrary length. Usually, the efficiency or running time of an algorithm is stated as a function relating the input length to the number of steps (time complexity) or storage location.

The aim of analyzing an algorithm is to discover its characteristics to evaluate its suitability for various applications or compare it with other algorithms for the same application. The characteristics of interests are most often the primary resource of time and space, particularly time. To simplify, it is important to understand how long the implementation of a particular algorithm will run on a particular computer and how much space it will require.

Choosing a suitable scheduling algorithm, scheduling is a fundamental operating system function. Almost all computer resources are scheduled before use. The CPU is one of the primary computer resources. Thus, its scheduling is central to operating system design. CPU scheduling determines which process runs when there are multiple run able processes. CPU scheduling is important because it can have a big effect on resource utilization and the overall performance of the system. CPU scheduling decides which process perform when their execution and acquire how many and which resources. CPU scheduling is essential because it performs a major role in efficient resource utilization and it positively affects the overall system performance. If there are multiple process ready for their execution, then it decides which process perform his execution and when. M. Akhtar, B. Hamid, I. ur-Rehman, M. Humayun, M. Hamayun and H. Khurshid (2015)

STATEMENT OF THE PROBLEM

In a single-processor system, only one process can run at a time, any other process must wait until the CPU is free and can be rescheduled. Multi programmed systems provide an environment in which various system resources (for example CPU, memory, and peripheral devices) are used. In Multiprogramming operating system, CPU scheduling plays a very important role. CPU scheduling deals with the problem to which process of the CPU should be allocated. For scheduling the processes in different ways, there are many different scheduling algorithms. However, each of these scheduling algorithms has its advantages and disadvantages. Hence, for resources to be utilized effectively there is a need to choose a suitable scheduling algorithm by operating system designers. Moreover, there are criteria for choosing a suitable scheduling algorithm such as throughput, CPU utilization, waiting time, response time, and turnaround time. Therefore, in this research, the main types of scheduling algorithms are compared together to determine the most suitable in terms of their average waiting time.

AIM OF THE STUD

The research aims to compare the three scheduling algorithms (i.e. First Come First Serve, Shortest Job First, and Round Robin Scheduling Algorithms) to determine the best among them in terms of their average waiting time.

OBJECTIVES OF THE STUDY

The primary objective of the research is to use the results of the comparison among the CPU scheduling algorithms to determine the most suitable in terms of speed.

SIGNIFICANT OF THE STUDY

This research, which is based on the comparative analysis of the various existing and most common types of scheduling algorithms for operating systems, will educate operating system designers to know which scheduling algorithm is most suitable for the different existing operating systems.

SCOPE AND LIMITATIONS

An operating system manages the allocation of resources and services such as memory, CPU and peripheral devices. However, in this research, I will look into how CPU resources are being scheduled in terms of average waiting time.

RELATED LITERATURE

As mentioned by the authors (Jayeshree, Somani & Chhatwani 2013) that, in Multiprogramming operating system, CPU scheduling plays a very important role. CPU scheduling deals with the problem to which process the CPU should be allocated. For scheduling the processes in different ways, there are many different scheduling algorithms. However, the main objective of their research was to compare the various scheduling algorithms like First-Come-First-Serve (FCFS) scheduling algorithm, Shortest Job First (SJF) scheduling algorithm, Priority scheduling algorithm, Round Robin (R-R) scheduling algorithm, Multilevel Queue scheduling algorithm, and Multilevel Feedback Queue scheduling algorithm. Moreover, they consider the following processes with their burst time and priority as follows:

Process	Burst time	Priority
P1	24	02
P2	03	01
P3	03	03

They also assume the time quantum to be 04ms. Below is a table showing the average waiting time they have calculated for FCFS, RR, SJF, and priority scheduling algorithm.

Algorithm

Algorithm	Avg Waiting time(ms)
FCFS	17
SJF	03
Priority	10
RR	5.66

Moreover, the following table shows a comparison of various scheduling algorithm on different parameters:

Sr. No.	Parameters	FCFS Algorithm	SJF Algorithm	Priority Algorithm	R-R Algorithm	Multilevel Queue Algorithm	Multilevel Feedback Queue Algorithm
1	Preemption	This scheduling algorithm is	This scheduling algorithm is preemptive.	This scheduling algorithm is	This scheduling algorithm is	This scheduling algorithm is	This scheduling algorithm is

		non-preemptive.		also preemptive.	also preemptive.	also preemptive.	also preemptive.
2	Complexity	This is simplest scheduling algorithm.	This algorithm is difficult to understand and code.	This algorithm is also difficult to understand.	In this scheduling algorithm, performance heavily depends upon the size of time quantum.	This algorithm is difficult to understand and code.	This algorithm is difficult to understand and code and its performance depends upon the size of time quantum.
3	Allocation	In this, it allocates the CPU in the order in which the process arrives.	In this, the CPU is allocated to the process with least CPU burst time.	It is based on the priority. The higher priority job can run first.	In this, the CPU is allocated in the order in which the process arrives but for fixed time slice.	In this, the CPU is allocated in the order in which the process arrives but for fixed time slice.	In this also, the CPU is allocated to the process of higher priority queue.
4	Waiting Time	In this, the average waiting time is large.	In this, the average waiting time is small as compared to FCFS scheduling algorithm.	In this, the average waiting time is small as compared to FCFS scheduling algorithm.	In this, the average waiting time is large as compared to all the three scheduling algorithms.	In this, the average waiting time is small as compared to FCFS scheduling algorithm.	In this, the average waiting time is small as compared to FCFS scheduling algorithm.

Also, they concluded that the first come first serve scheduling algorithm is simple to understand and suitable only for the batch system where the waiting time is large. The shortest job first scheduling algorithm deals with a different approach. In this algorithm, the major benefit is that it gives the minimum average waiting time. The priority scheduling algorithm is based on the priority in which the highest priority job can run first and the lowest priority job needs to wait though it will create a problem of starvation. The round-robin scheduling algorithm is pre-emptive which is based on FCFS policy and time quantum. This algorithm is suitable for time-sharing systems. In multilevel queue scheduling, processes are permanently assigned to a queue depending upon its nature and no process in the lower priority queue could run unless the higher priority queues were empty. Also, it is pre-emptive in nature. Multilevel feedback queue scheduling is also pre-emptive in nature and it allows the processes to move between the queues depending upon the given time quantum.

As mentioned by the authors (Yaashuwanth &Ramesh, 2009) that, the scheduling algorithms play a significant role in the design of real-time embedded systems. Simple round robin architecture is not efficient to be implemented in embedded systems because of higher context switch rate, larger waiting time and larger response time. Missing deadlines will degrade the system performance in soft real-time systems. The main objectives of their research are to develop the scheduling algorithm which removes the drawbacks in simple round-robin architecture. A comparison with round-robin architecture to the proposed architectures has been made. It is observed that the proposed architectures solve the problems encountered in round-robin architecture in soft real-time by decreasing the number of context switches waiting time and response time thereby increasing the system throughput. However, the proposed (that is the shortest round-robin) architecture eliminates the defects of implementing a simple round-robin architecture by scheduling of processes based on the CPU burst, A dedicated small processor used to reduce the burden of the main processor is assigned for rearranging of processes in the ascending order based on the CPU burst of the process (lower to higher) The proposed architecture has greater waiting time response time and throughput thereby improving the system performance. Besides, the other proposed architecture (that is intelligent time slice for round-robin) eliminates the defects of implementing simple round-robin architecture in the soft real-time system by introducing a concept called intelligent time-slicing which depends on, three aspects they are the priority, average CPU burst, and context switch avoidance time.

They also compare simple round-robin, shortest round-robin, and intelligent round-robin by considering the following processes with their burst time;

Process ID	CPU burst time (milliseconds)	Priority
1	25	2

2	5	3
3	15	1
4	8	2
5	10	1

Assume time slice = 04ms

TABULAR COMPARISON BETWEEN ROUND ROBIN, SHORTEST ROUND ROBIN AND INTELLIGENT TIME SLICE FOR ROUND ROBIN

Algorithm	Waiting time in milliseconds	Turn around time in milliseconds
Simple round robin	31	44
Shortest round robin	22	36
Intelligent time slice for round robin	25	37

A comparative study of round-robin architecture, shortest round-robin and intelligent time slice for round-robin architecture are made. They concluded that the proposed architectures are superior as it has less waiting, response times, usually less pre-emption and context switching thereby reducing the overhead and saving of memory space.

As mentioned by the authors (AnkurBhardway, Singh & Gaurav, 2013) that scheduling algorithms deals with the problem and decides which problem should be executed next and allocate to the CPU. However, they compare the following scheduling algorithm FCFS, SJF, Priority algorithm and Round Robin Algorithm based on the following parameters namely; pre-emption, complexity, Allocation, waiting time, usability, type of system. Below is a table showing their comparison:

Sr.no	Parameter	FCFS Algorithm	SJF Algorithm	Priority algorithm	Round Robin Algorithm
1	Preemption	This scheduling algorithm is non-preemptive.	This scheduling algorithm is preemptive.	This scheduling algorithm is also preemptive.	This scheduling algorithm is also preemptive.
2	Complexity	This is simplest scheduling algorithm.	This algorithm is difficult to understand and code.	This algorithm is also difficult to understand.	In this scheduling algorithm performance heavily depends upon the size of time quantum.
3	Allocation	In this scheduling algorithm it allocates the CPU in the order in which the processes arrives.	In this scheduling algorithm CPU is allocated to the process with least CPU burst time.	This scheduling algorithm is based on priority the higher priority job can run first.	In this scheduling algorithm the preemption take place after a fixed interval of time.
4	Application	This scheduling algorithm is good for non-	This scheduling algorithm is also good for non-	This scheduling algorithm is also good for non-	This scheduling algorithm is good for

		interactive system.	interactive system.	interactive system.	interactive system.
5	Waiting Time	In this scheduling algorithm the Average waiting time is large.	In this scheduling algorithm the Average waiting time is small as compare to FCFS scheduling algorithm.	In this scheduling algorithm the Average waiting time is small as compare to FCFS scheduling algorithm.	In this scheduling algorithm the Average waiting time is large as compare to all the three scheduling algorithm.
6	Usability	This scheduling algorithm is never recommended whenever performance is a major issue.	In this scheduling algorithm the problem is to know the length of time for which the CPU is needed by the process.	This scheduling algorithm is the sometime becomes the biggest cause of starvation.	In this scheduling algorithm if the quantum size is large then this algorithm become same as FCFS algorithm and its performance degrade.
7	Type of system	This scheduling algorithm is suitable for Batch system.	This scheduling algorithm is also suitable for Batch system.	This scheduling algorithm is based upon priority.	This scheduling algorithm is suitable for time sharing system.

Also, they concluded that the first come first serve scheduling algorithm is simple to understand and suitable only for the batch system where the waiting time is large. The shortest job first scheduling algorithm deals with a different approach in this algorithm the major benefit is it gives the minimum average waiting time. The priority scheduling algorithm is based on the priority in which the highest priority job can run first and the lowest priority job needs to wait though it will create a problem of starvation. The round-robin scheduling algorithm is pre-emptive which is based on round-robin policy one of the scheduling algorithms which follows the interactive system and the round-robin scheduling algorithm deal with the time-sharing system.

As mentioned by the authors (Neetu, Goel &Garg, 2012) that, Developing CPU scheduling algorithms and understanding their impact in practice can be difficult and time consuming due to the need to modify and test operating system kernel code and measure the resulting performance on a consistent workload of the real application. As the processor is important resources, CPU scheduling becomes very important in accomplishing the operating system (OS) design goals. The intention is to allow as many as possible running processes at all times to make the best use of CPU. Their research aimed to compare various scheduling algorithms for a single CPU and show which algorithm is best for a particular situation. Moreover, the consider the following set of processes with the length of the CPU burst time in milliseconds as shown in the table below;

Process ID	Burst Time(ms)	Priority
P0	12	3
P1	2	1
P2	3	3
P3	2	4
P4	6	2

Below is a table showing their comparison of scheduling algorithms in terms of waiting time and turnaround time

Process ID	Turnaround Time (ms)			
	FCFS	SJF	Round Robin	Priority

P0	12	25	25	20
P1	14	2	7	2
P2	17	7	10	23
P3	19	4	12	25
P4	25	13	23	8
Avg Turnaround Time	17.4	10.2	15.4	15.6

Turnaround time for individual process and average turnaround time for each schedule

Process ID	Waiting Time (ms)			
	FCFS	SJF	Round Robin	Priority
P0	0	13	13	8
P1	12	0	5	0
P2	14	4	7	20
P3	17	2	10	23
P4	19	7	17	2
Avg Waiting Time	12.4	5.2	10.4	10.6

Waiting time for individual process and average waiting time for each schedule

Hence, they concluded that the treatment of the shortest process in SJF scheduling tends to result in increased waiting time for long processes. And the long process will never get served, though it produces minimum average waiting time and average turnaround time. It is recommended that any kind of simulation for any CPU scheduling algorithm has limited accuracy. The only way to evaluate a scheduling algorithm is to code it and put it in the operating system, only then a proper working capability of the algorithm can be measured in real-time systems.

As mentioned by the authors (Singh, Vinod & Anjanipandey, 2013) that Scheduling is a fundamental operating system function since almost all computer resources are scheduled before use. The CPU is one of the primary computer resources. Central Processing Unit (CPU) scheduling plays an important role by switching the CPU among various processes. A processor is an important resource in a computer; the operating system can make the computer more productive. The purpose of the operating system is to allow the process as many as possible running at all the time to make the best use of CPU. The high efficient CPU scheduler depends on the design of the high-quality scheduling algorithms which suits the scheduling goals. They reviewed various fundamental CPU scheduling algorithms for a single CPU and shows which algorithm is best for a particular situation. Moreover, they have considered the following process with burst time and priority;

Process ID	Burst time (ms)	Priority
P1	10	3
P2	2	1
P3	8	4
P4	6	2

Besides, below is a table showing the comparison between FCFS, SJF, RR, and priority scheduling in terms of waiting time and turnaround time.

Process ID	Waiting Time (ms)			
	FCFS	SJF	Round Robin	Priority
P1	0	16	12	8
P2	10	0	5	0
P3	12	8	17	18
P4	20	2	20	2
Avg Waiting Time	10.5	6.5	13.5	7

Process ID	Turnaround Time (ms)			
	FCFS	SJF	Round Robin	Priority
P1	10	26	22	18
P2	12	2	7	2
P3	20	16	25	26
P4	26	8	26	8
Avg Turnaround Time	17	13	20	13.5

Hence, they concluded that the SJF scheduling algorithm is to serve all types of jobs with optimum scheduling criteria. The treatment of the shortest process in SJF scheduling tends to result in increased waiting time for long processes. And the long process will never get served, though it produces minimum average waiting time and average turnaround time. The shortest job first scheduling algorithm deals with a different approach, in this algorithm; the major benefit is it gives the minimum average waiting time. It is recommended that any kind of simulation for any CPU scheduling algorithm has limited accuracy. The only way to evaluate a scheduling algorithm to code it and has to put it in the operating system, only then a proper working capability of the algorithm can be measured in real-time system

In this research, our approach is similar to the above mentioned authors, but distinct by focusing on the average waiting time of all the scheduling algorithms for the comparison as against the authors mentioned in the literature.

METHODOLOGY

SCHEDULLING ALGORITHMS

CPU Scheduling is a process of determining which process will own CPU for execution while another process is on hold. The main task of CPU scheduling is to make sure that whenever the CPU remains idle, the OS at least select one of the processes available in the ready queue for execution. The selection process will be carried out by the CPU scheduler. It selects one of the processes in memory that are ready for execution.

OS may feature up to 3 distinct types of schedulers: a long-term scheduler (also known as an admission scheduler or high-level scheduler), a mid-term or medium-term scheduler and a short-term scheduler (also known as a dispatcher or CPU scheduler).

ISSN 2229-5518

A. Long-term Scheduler

The long-term or admission scheduler decides which jobs or processes are to be admitted to the ready queue; that is, when an attempt is made to execute a process its admission to the set of currently executing processes is either authorized or delayed by the long-term scheduler. Thus, this scheduler dictates what processes are to run on a system, and the degree of concurrency to be supported at any one time.

B. Mid-term Scheduler

The mid-term scheduler temporarily removes process from main memory and place them on secondary memory (such as a disk drive) or vice versa. This is commonly referred to as “swapping of processes out" or "swapping in" (also incorrectly as "paging out" or "paging in").

C. Short-term Scheduler

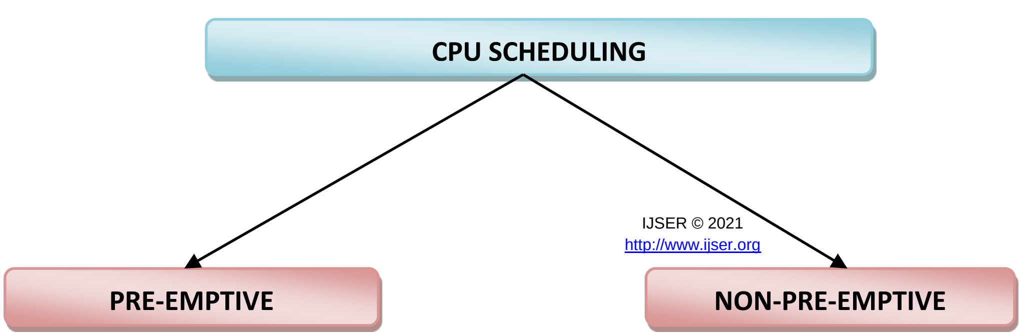
The short-term scheduler (also known as the CPU scheduler) decides which of processes in the ready queue, in memory are to be executed (allocated a CPU) next following a clock interrupt, an Input-Output (IO) interrupt and an OS call or another form of signal.

Thus, the short-term scheduler makes scheduling decisions much more frequent than the long-term or mid-term schedulers. This scheduler can be pre-emptive, implying that it is capable of forcibly removing processes from a CPU when it decides to allocate that CPU to another process, or non pre-emptive (also known as "voluntary" or "co-operative"), in that case the scheduler is unable to force processes off the CPU. NeetuGoel,R.B. Garg 2012

The Purpose of a Scheduling algorithm

1. Maximum CPU utilization
2. Fare allocation of CPU
3. Maximum throughput
4. Minimum turnaround time
5. Minimum waiting time
6. Minimum response time

Types of CPU Scheduling



ISSN 2229-5518

Fig. 1: kinds of scheduling methods:

Source: CPU Scheduling Algorithms in Operating Systems,https://www.guru99.com/cpu-scheduling-algorithms.html

Pre-emptive Scheduling

In Preemptive Scheduling, the tasks are mostly assigned with their priorities. Sometimes it is important to run a task with a higher priority before another lower priority task, even if the lower priority task is still running. The lower priority task holds for some time and resumes when the higher priority task finishes its execution.

Non Pre-emptive Scheduling

In this type of scheduling method, the CPU has been allocated to a specific process. The process that keeps the CPU busy will release the CPU either by switching context or terminating. It is the only method that can be used for various hardware platforms. That's because it doesn't need special hardware (for example, a timer) like preemptive scheduling.

Dispatcher

- The dispatcher is the module that gives control of the CPU to the process selected by the scheduler. This function involves:
    - Switching context.
    - Switching to user mode.
    - Jumping to the proper location in the newly loaded program.
- The dispatcher needs to be as fast as possible, as it is run on every context switch. The time consumed by the dispatcher is known as dispatch latency. Abraham Silberschatz, Greg Gagne, and Peter Baer Galvin,

Starvation

This is a problem encountered in multitasking where a process is perpetually denied necessary resources. Without those resources, the program can never finish its task. Starvation is usually caused by an overly simplistic scheduling algorithm. For example, if a (not very well designed) multi-tasking system always switches between the first two tasks while a third never gets to run, then the third task is being starved of CPU time. The scheduling algorithm, which is part of the kernel, is supposed to

ISSN 2229-5518

allocate resources equitably; that is, the algorithm should allocate resources so that no process perpetually lacks necessary resources. Y. A. Adekunle, Z. O. Ogunwobi, A. Sarumi, B.T. Efuwape, S. Ebiesuwa, and Jean-Paul Ainam, 2014

Fairness

This is the criterion that ensures that each process has an equal chance of being executed as the other. With fairness brought to the fore no particular process has a so-called preference or unfair advantage over the others.Y.A.Adekunle, Z. O. Ogunwobi, A. Sarumi, B.T. Efuwape, S. Ebiesuwa, and Jean-Paul Ainam, 2014

CPU SCHEDULING CRITERIA

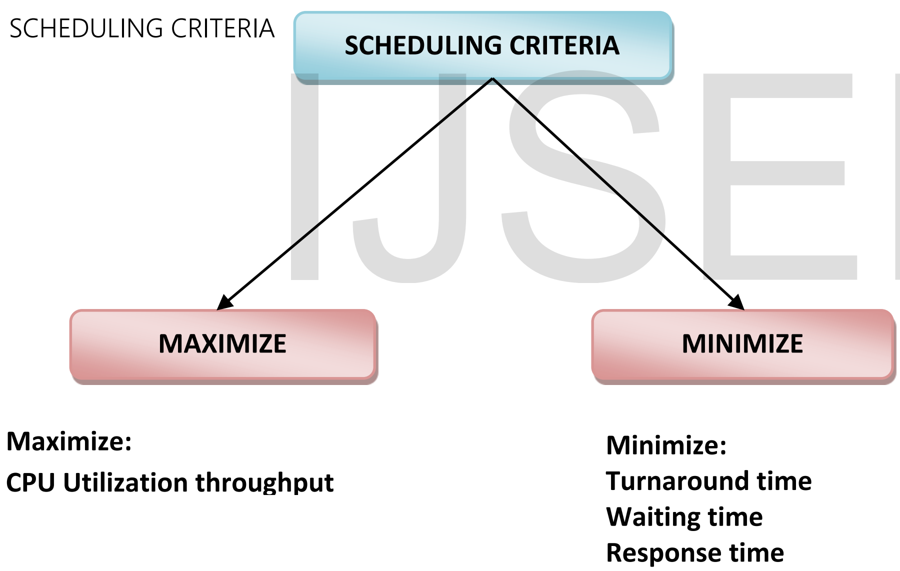


Fig. 2: CPU scheduling criteria

Source: CPU Scheduling Algorithms in Operating Systems,https://www.guru99.com/cpu-scheduling-algorithms.html

Different CPU scheduling algorithms have different properties, and the choice of a particular algorithm may favor one class of processes over another. In choosing which algorithm to use in a particular situation, we must consider the properties of the various algorithms. Many criteria have been suggested for comparing CPU scheduling algorithms. Which characteristics are used for comparison can make a substantial difference in which algorithm is judged to be best *(NeetuGoel,R.B. Garg, 2012)*. The criteria include the following:

Maximize:

CPU utilization: CPU utilization is the main task in which the operating system needs to make sure that CPU remains as busy as possible. It can range from 0 to 100 percent. However, for the RTOS, it can be range from 40 percent for low-level and 90 percent for the high-level system.

Throughput: The number of processes that finish their execution per unit time is known Throughput. So, when the CPU is busy executing the process, at that time, work is being done, and the work completed per unit time is called Throughput.

Minimize:

Waiting time: Waiting time is an amount that specific process needs to wait in the ready queue.

Response time: It is an amount to time in which the request was submitted until the first response is produced.

Turnaround Time: Turnaround time is an amount of time to execute a specific process. It is the calculation of the total time spent waiting to get into the memory, waiting in the queue and, executing on the CPU. The period between the time of process submission to the completion time is the turnaround time.

Types of CPU scheduling Algorithm

There are mainly six types of process scheduling algorithms

1. First Come First Serve (FCFS)
2. Shortest-Job-First (SJF) Scheduling
3. Shortest Remaining Time

Source: Operating System Scheduling algorithms – Tutorialspoint, https://www.tutorialspoint.com/operating_system/os_process_scheduling_algorithms.htm

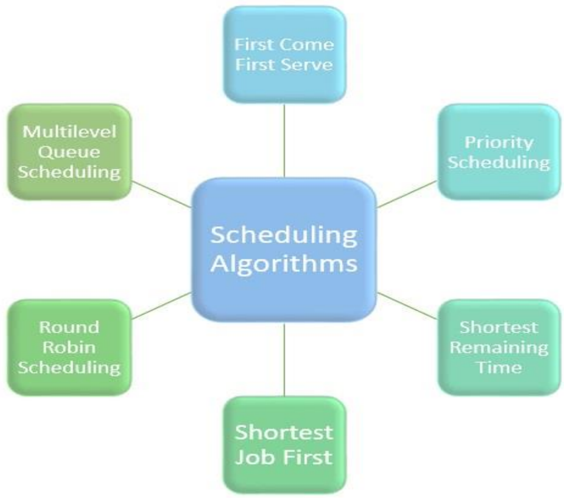


Fig. 3: Types of CPU scheduling algorithms

Source: CPU Scheduling Algorithms in Operating Systems,https://www.guru99.com/cpu-scheduling-algorithms.html

THE SCHEDULING ALGORITHMS

1. THE FIRST-COME FIRST-SERVED (FCFS) SCHEDULING ALGORITHM

This is one of the simplest and oldest scheduling algorithms. Usually, a single queue is used to gather the jobs that request system resources. Every time a new job is scheduled, it is enlisted at the end of the queue. When resources are available they are allocated to the job in the first position of the queue and the job is removed. Any time resources are released they are assigned to the new head of the queue. The FCFS algorithm is not pre-emptive and once a process gets the resources, it runs until completion. The implementation of this scheduling strategy is usually straightforward and the CPU overhead introduced is minimal, especially if the algorithm is implemented using a FIFO queue.

2. SHORTEST JOB FIRST SCHEDULING ALGORITHM

In this scheduling algorithm, the CPU is allocated to the process which has the smallest next CPU burst. The SJF uses the FCFS to break the tie (a situation where two processes have the same length next CPU burst). The SJF algorithm can be pre-emptive or non-pre-emptive. In pre-emptive SJF scheduling, the execution of a process that is currently running is interrupted to give the CPU resources to a newly arrived process with a shorter next CPU burst. On the other hand, the non-pre-emptive SJF will allow the currently running process to finish its CPU burst before a new process is allocated to the CPU.

3. ROUND ROBIN SCHEDULING ALGORITHM

ISSN 2229-5518

Round robin scheduling is a pre-emptive version of first-come-first-served scheduling. Processes are dispatched on a first-in-first-out sequence but each process is allowed to run for only a limited amount of time. This time interval is known as a time-slice or time quantum. In this, the ready queue is treated as the circular queue. One of the two things will happen in Round-Robin. Firstly, the process may have burst-time less than or equal to time quantum. In this case, the process will execute and after completion release the CPU by itself. Secondly, the process may have burst time greater than time quantum. In this case, the process will execute for 1-time quantum and then it is pre-empted. Then context-switch will be executed and the CPU scheduler will select the next process to execute. The preempted process will be put at the tail of the ready queue. This continues until the execution of all the processes is complete

SYSTEM ANALYSIS

4.0 RESULT OF COMPARISON

This phase will be used to compare three scheduling algorithms in terms of their average waiting time with the aid of the java programs developed.

Comparison of the three scheduling algorithm:

They three scheduling algorithm that will be compare are First Come First Serve, Round Robin and Shortest Job First Scheduling Algorithm. However, the comparison will involve three cases and they will be compare base on their waiting time.

Case 1: Increasing order, this is a situation whereby the burst time of the jobs increases simultaneously.

Case 2: Decreasing order, this is a situation whereby the burst time of the jobs decreases simultaneously.

Case 3: Random order, this is a situation whereby the burst time of the jobs is random.

Comparison of First Come First Serve, Round Robin and Shortest Job First Scheduling Algorithm.

NOTE: RR-Round Robin, FCFS-First Come First Serve, SJF-Shortest Job First, WT-Waiting Time.

4.1 Result of Comparison

For Ten (10) Jobs:

The time slice for RR1 is 5 and RR2 is 10.

Case 1: Increasing order: suppose we have ten jobs of burst time 1, 2,3,4,5,6,7,8,9,10.

FCFS

JOB	BT	WT
JOB1	1	0
JOB2	2	1
JOB3	3	3
JOB4	4	6
JOB5	5	10
JOB6	6	15
JOB7	7	21
JOB8	8	28
JOB9	9	36
JOB10	10	45

THE TOTAL WAITING TIME IS : 165.0

THE AVERAGE WAITING TIME IS : 16.5

BUILD SUCCESSFUL (total time: 17 seconds)

RR1

JOB	BT	WT
JOB1	1	0
JOB2	2	1
JOB3	3	3
JOB4	4	6
JOB5	5	10
JOB6	6	35
JOB7	7	36
JOB8	8	38
JOB9	9	41
JOB10	10	45

THE TOTAL WAITING TIME IS 215.0

THE AVERAGE WAITING TIME IS 21.5

BUILD SUCCESSFUL (total time: 26 seconds)

RR2

process	BT	WT
process1	1	0
process2	2	1
process3	3	3
process4	4	6
process5	5	10
process6	6	15
process7	7	21
process8	8	28
process9	9	36
process10	10	45

The total waiting time is : 165.0

Average waiting time : 16.5

BUILD SUCCESSFUL (total time: 12 seconds)

SJF

JOB	BT	WT
JOB1	1	0
JOB2	2	1
JOB3	3	3
JOB4	4	6
JOB5	5	10
JOB6	6	15
JOB7	7	21
JOB8	8	28
JOB9	9	36
JOB10	10	45

THE TOTAL WAITING TIME IS :165.0

THE AVERAGE WAITING TIME IS :16.5

BUILD SUCCESSFUL (total time: 14 seconds)

From the above result you can see that we have two RR's, one is RR1 which has time slice 5 and average waiting time 21.5 while the RR2 has time slice 10 and average waiting time 16.5. Looking at the result we can see that FCFS, RR2 and SJF have 16.5 as their average waiting time. However, from the above result we can say that Round Robin produced minimum average waiting time whenever the time quantum is greater than or equal to the highest burst time.

Case 2: Decreasing order: suppose we have ten jobs with burst time 10, 9, 8, 7, 6, 5, 4, 3, 2, 1.

FCFS

RR1

JOB	BT	WT
JOB1	10	0
JOB2	9	10
JOB3	8	19
JOB4	7	27
JOB5	6	34
JOB6	5	40
JOB7	4	45
JOB8	3	49
JOB9	2	52
JOB10	1	54

THE TOTAL WAITING TIME IS : 330.0
THE AVERAGE WAITING TIME IS : 33.0
BUILD SUCCESSFUL (total time: 14 seconds)

JOB	BT	WT
JOB1	10	35
JOB2	9	40
JOB3	8	44
JOB4	7	47
JOB5	6	49
JOB6	5	25
JOB7	4	30
JOB8	3	34
JOB9	2	37
JOB10	1	39

THE TOTAL WAITING TIME IS 380.0
THE AVERAGE WAITING TIME IS 38.0
BUILD SUCCESSFUL (total time: 23 seconds)

RR2

JOB	BT	WT
JOB1	10	0
JOB2	9	10
JOB3	8	19
JOB4	7	27
JOB5	6	34
JOB6	5	40
JOB7	4	45
JOB8	3	49
JOB9	2	52
JOB10	1	54

THE TOTAL WAITING TIME IS 330.0
THE AVERAGE WAITING TIME IS 33.0
BUILD SUCCESSFUL (total time: 23 seconds)

SJF

JOB	BT	WT
JOB10	1	0
JOB9	2	1
JOB8	3	3
JOB7	4	6
JOB6	5	10
JOB5	6	15
JOB4	7	21
JOB3	8	28
JOB2	9	36
JOB1	10	45

THE TOTAL WAITING TIME IS :165.0
THE AVERAGE WAITING TIME IS :16.5
BUILD SUCCESSFUL (total time: 17 seconds)

Case 3: Random Order: suppose we have ten jobs with burst time 8, 1, 3, 7, 10, 2, 4, 9, 6, 5.

FCFS

JOB	BT	WT
JOB1	10	0
JOB2	1	8
JOB3	3	9
JOB4	7	12
JOB5	10	19
JOB6	2	29
JOB7	4	31
JOB8	9	35
JOB9	6	44
JOB10	5	50

THE TOTAL WAITING TIME IS : 237.0
THE AVERAGE WAITING TIME IS : 23.7
BUILD SUCCESSFUL (total time: 28 seconds)

RR1

JOB	BT	WT
JOB1	8	35
JOB2	1	5
JOB3	3	6
JOB4	7	38
JOB5	10	40
JOB6	2	19
JOB7	4	21
JOB8	9	45
JOB9	6	49
JOB10	5	35

THE TOTAL WAITING TIME IS 293.0
THE AVERAGE WAITING TIME IS 29.3
BUILD SUCCESSFUL (total time: 23 seconds)

RR2

JOB	BT	WT
JOB1	8	0
JOB2	1	8
JOB3	3	9
JOB4	7	12
JOB5	10	19
JOB6	2	29
JOB7	4	31
JOB8	9	35
JOB9	6	44
JOB10	5	50

THE TOTAL WAITING TIME IS 237.0
THE AVERAGE WAITING TIME IS 23.7
BUILD SUCCESSFUL (total time: 23 seconds)

SJF

JOB	BT	WT
JOB2	1	0
JOB6	2	1
JOB3	3	3
JOB7	4	6
JOB10	5	10
JOB9	6	15
JOB4	7	21
JOB1	8	28
JOB8	9	36
JOB5	10	45

THE TOTAL WAITING TIME IS :165.0
THE AVERAGE WAITING TIME IS :16.5
BUILD SUCCESSFUL (total time: 27 seconds)

4.2 Gantt chart for Ten Jobs:

Increasing Order:

FCFS

P1	P2	P3	P4	P5	P6

0 1 3 6 10 15

P7	P8	P9	P10

21 28 36 45 55

RR1 Time Slice = 5.

P1	P2	P3	P4	P5	P6

0 1 3 6 10 15

P7	P8	P9	P10	P6	P7	P8

20 25 30 35 40 41 43

P9	P10

46 50 55

RR2 Time Slice>=10

ISSN 2229-5518

P1	P2	P3	P4	P5	P6

0 1 3 6 10 15

P7	P8	P9	P10

21 28 36 45 55

SJF

P1	P2	P3	P4	P5	P6

0 1 3 6 10 15

P7	P8	P9	P10

21 28 36 45 55

Decreasing Order:

FCFS

P1	P2	P3	P4	P5	P6

0 10 19 27 34 40

P7	P8	P9	P10

45 49 52 54 55

RR1 Time Slice =5

P1	P2	P3	P4	P5	P6

0 5 10 15 20 25

P7	P	P9	P10	P1	P2	P3	P4	P5

29 32 34 35 36 41 45 48 50 51

RR2 Time Slice>=10

P1	P2	P3	P4	P5	P6

ISSN 2229-5518

0 10 19 27 34 40

P7	P8	P9	P10

45 49 52 54 55

SJF

P10	P9	P8	P7	P6	P5	P4	P3

0 1 3 6 10 15 21 28

P2	P1

36 45 55

Random Order:

FCFS

P1	P2	P3	P4	P5	P6	P7	P8

0 8 9 12 19 29 31 35

P9	P10

44 50 55

RR1 Time Slice = 5

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P1

0 5 6 9 14 19 21 25 30 35 40

P4	P5	P8	P1

43 45 50 54 55

RR2 Time Slice>=10

P1	P2	P3	P4	P5	P6	P7	P8

0 8 9 12 19 29 31 35

P9	P10	
44	50	55

SJF

P2	P6	P3	P7	P10	P9	P4	P1	P8
0	1	3	6	10	15	21	28	36

P5	
45	55

4.3 Summary of the result:

For Tens Jobs: Increasing Order

Scheduling Algorithm	Total Waiting Time	Average Waiting Time
FCFS	165.0	16.5
RR	215.0	21.5
SJF	165.0	16.5

For Twenty Jobs: Increasing Order

Scheduling Algorithm	Total Waiting Time	Average Waiting Time
FCFS	1330.0	66.5
RR	1780.0	89.0
SJF	1330.0	66.5

For Thirty Jobs: Increasing Order

Scheduling Algorithm	Total Waiting Time	Average Waiting Time

FCFS	4493.0	149.8
RR	6069.0	202.3
SJF	4495.0	149.8

For Forty Jobs: Increasing Order

Scheduling Algorithm	Total Waiting Time	Average Waiting Time
FCFS	10660.0	266.5
RR	14460.0	361.5
SJF	10660.0	266.5

For Fifty Jobs: Increasing Order

Scheduling Algorithm	Total Waiting Time	Average Waiting Time
FCFS	20825.0	416.5
RR	28325.0	566.5
SJF	20825.0	416.5

For Ten Jobs: Decreasing Order

Scheduling Algorithm	Total Waiting Time	Average Waiting Time
FCFS	330.0	33.0
RR	380.0	38.0
SJF	165.0	16.5

For Twenty Jobs: Decreasing Order

Scheduling Algorithm	Total Waiting Time	Average Waiting Time
FCFS	2660.0	133.0
RR	3110.0	155.0
SJF	1330.0	66.5

For Thirty Jobs: Decreasing Order

Scheduling Algorithm	Total Waiting Time	Average Waiting Time
FCFS	8990.0	299.6
RR	10566.0	352.2
SJF	4495.0	149.8

For Forty Jobs: Decreasing Order

Scheduling Algorithm	Total Waiting Time	Average Waiting Time
FCFS	21320.0	533.0
RR	25120.0	628.0
SJF	10660.0	266.5

For Fifty Jobs: Decreasing Order

Scheduling Algorithm	Total Waiting Time	Average Waiting Time
FCFS	41650.0	833.0
RR	49150.0	983.0

SJF	20825.0	416.5

For Ten Jobs: Random Order

Scheduling Algorithm	Total Waiting Time	Average Waiting Time
FCFS	237.0	23.7
RR	293.0	29.3
SJF	165.0	16.5

For Twenty Jobs: Random Order

Scheduling Algorithm	Total Waiting Time	Average Waiting Time
FCFS	2252.0	112.6
RR	2632.0	131.6
SJF	1330.0	66.5

For Thirty Jobs: Random Order

Scheduling Algorithm	Total Waiting Time	Average Waiting Time
FCFS	6869.0	228.9
RR	8478.0	282.6
SJF	4495.0	149.8

For Forty Jobs: Random Order

Scheduling Algorithm	Total Waiting Time	Average Waiting Time

FCFS	17818.0	445.7
RR	21396.0	534.9
SJF	10660.0	266.5

For Fifty Jobs: Random Order

Scheduling Algorithm	Total Waiting Time	Average Waiting Time
FCFS	36398.0	727.9
RR	40849.0	816.9
SJF	20825.0	416.5

DISCUSSION AND CONCLUSION

5.1 DISCUSSION

As clearly seen from the result, Shortest Job First has the lowest average waiting time and the outcome of shortest job first is the same in all the three cases. Round Robin has been seen in two perspectives. The first perspective is when the time quantum is less than the highest burst time of the process and this result in the maximum average burst time. The second perspective is when the time quantum is greater than or equal to the highest burst time of the process and this produce same result as that of first come first serve.

However, the above discussion is based on the result obtained from testing of 10, 20, 30, 40, and 50 processes in three different cases (i.e. increasing order, decreasing order and random order).

5.2 CONCLUSION

A comparison was made between three scheduling algorithms (i.e. first come first serve, Round Robin, and Shortest Job First) in terms of their average waiting time. From the results, the findings suggest that the Shortest Job First is most suitable scheduling algorithm as it possesses the lowest average waiting time.

5.3 RECOMMENDATION

As seen from the summary of the results, a test of 10, 20, 30, 40, and 50 processes was carried out using three different cases. As a recommendation for further study on this topic, more algorithms with greater values and more comprehensive test threshold values should be tested in order to acquire further knowledge on the problem of the topic.

Acknowledgment

This work was sponsored by a grant from the Tertiary Education Trust Fund (TETfund) Nigeria. I wish to thank my able Rector, Chief Librarian and Dean School of Science and above all the reviewer.

BIBLIOGRAPHY AND REFERENCE

Abraham.S.,Peter.B.G., Greg .G.(2012).Ninth Edition,operating System Concept, Chapter: Scheduling algorithm, page 273

Ankur B., Racchhpal S., &Gaurav. (2013). "Comparative Study of Scheduling Algorithms in Operating System, ISSN:2278-5183, vol, No 3, Issue1, April-May.

C. Yaashuwath and R. Rameh. (2009) ."A New Scheduling Algorithms for Real-Time Tasks" vol 6, No 2

CPU Scheduling Algorithms in Operating Systems, https://www.guru99.com/cpu-scheduling-algorithms.html

Harshita.J.,Subrata.C.,&Ramya.G.(2017).Survey on various Scheduling Algorithms, mperial Journal of Interdisciplinary Research (IJIR) Vol-3, Issue-5,ISSN: 2454-1362, http://www.onlinejournal.in

Jayashree S.S. and Pooja K.C. (2013) ."Comparative Study of Different CPU Scheduling Algorithms" ISSN:2820-088x, IJCSMC, Vol 2, Issue 11, November, pg.310-318.

Muhammad .A.,Bushra.H.,Inayat.R.,Mamoona.H., Maryam.H., &Hira.K.An Optimized Shortest job first Scheduling Algorithm for CPU Scheduling, J. Appl. Environ. Biol. Sci., 5(12)42-46(2015), SSN: 2090-4274 Journal of Applied Environmental and Biological Sciences

Neetu.G.,R.B. Garg|A2012).Comparative Study of CPU Scheduling Algorithms, International Journal of Graphics & Image Processing |Vol 2|issue 4|November2012 (IJGIP)

Operating System Scheduling algorithms Tutorialspoint, https://www.tutorialspoint.com/operating_system/os_process_scheduling_algorithms.htm

Pushpraj S., Vinod S. and Anjani P. (2014) ."Analysis and Comparison of CPU Scheduling Algorithm" ISSN:2250-2459, Volume 4, Issue 1, January.

Y.A.Adekunle, Z.O.Ogunwobi, A.Sarumi Jerry, B.T. Efuwape, SeunE., &Jean- P.A.,(2014).A Comparative Study of Scheduling Algorithms for Multiprogramming in Real-Time Systems, International Journal of Innovation and Scientific Research ISSN 2351-8014 Vol. 12No. 1Nov. 2014, pp. 180-185© 2014Innovative Space of Scientific Research Journalshttp://www.ijisr.issr-journals.org/