Comparative Analysis of Basic CPU Scheduling Algorithms

Sajida Fayyaz[1], Hafiz Ali Hamza[2], Saria Moin U Din[3] and Ehatsham Riaz[4]
[1-4]Department of Computer Science, University of Lahore (Sargodha Campus)
[1]sajida.fayyaz@cs.uol.edu.pk, [2]alihamzagondal12@yahoo.com, [3]saira.moin@cs.uol.edu.pk, [4]ehatshamriaz@gmail.com

***Abstract*– Computer science is all about processes and commands running parallel. In computer, the processor CPU (Central Processing Unit) may contain different cores or single core and one core handle one process at a time. Running of processes in parallel pattern may reduce the context switching because while process running in CPU interrupt will occur. During context switching CPU switch to other process and put the current process in waiting queue the time required for CPU to save the current process and load the next process is context switching time. While loading the next process for execution from ready queue the CPU required some sort of algorithms on the basis of which system decide. We include the discussion on such scheduling algorithms with respect to their response time, wait time and turnaround time. The objective of this paper is to examine the all CPU scheduling algorithms including First come First Serve (FCFS), Shortest Job First (SJF), Priority and Round Robin (RR) algorithms. After inspecting the simulation result using number of examples, we have to select the best algorithm for CPU scheduling. As main purposes of scheduling is to keep CPU busy every Jiff of a second so that processes don't have to wait much longer. While talking about Round Robin algorithms we have three different approaches working, which are Round Robin with (FCFS, SJF and Priority). We will also examine Round Robin with all these approaches and figure out which approach for Round Robin algorithm work more efficiently and did maximum utilization of CPU. Main motive of this paper is to maximize CPU utilization and decrease the average wait time and average turnaround time so we have to find the best serving algorithm to achieve this goal.**

***Keywords*– CPU Scheduling, Process, Scheduling Algorithms, Comparison, Wait Time, Burst Time, Gant Chart Turnaround Time and Response Time**

I. INTRODUCTION

In present time modern system use many resources like one or more processor, input output devices, main memory and many other system resources which combined to form a complex system. These complex resources can have to manage through some supervisor which can control and manage all the resources of Computer. This supervisor is operating system (OS). In [7] Andrew explores that OS run processes in two different modes that are kernel mode and user mode, operating system tasks includes the allocation of resources to different processes. Processes are actually programs in running form. They normally started by user or system itself. Allocation of resources means that there are some processes who want to use the CPU to complete their task so operating systems allocate them CPU one by one.

In present time systems don't use traditional CPU programming but multiprogramming technique. In [2] M. Sindhu defined that the CPU is unique and crucial computer resource CPU consist of different cores on the basis of these number of cores the processors are categorized. Every core have some nonvolatile memory attached to it for faster access of data called caches memory. When we talk about process it contains many threads (light wait process) a thread cannot exist outside a process. Thread example includes that when we run a simple Microsoft word application it perform number of task at the back end, while we are typing it simultaneously checking the spellings and also auto-saving document etc. if we don't have this multithreading concept then in above example while system is saving the document it is unable to read key strokes.

After creation a process can be in different states which are new, running, waiting, ready and terminated. When a process created it's in new state, when the process is being executed it's in running form, when it is waiting for some reason it's in waiting state and when process is ready and waiting for CPU at that time process is in ready state and when process had completed its task and exit the CPU it's in terminated state. There are several queues that separate out the processes from each other are job queue, waiting queue and ready queue in job queue all the processes which are created rest there. Some processes have to wait for some event to happen it can be any resource allocation just in case if resource is busy with any other process. These processes remain in waiting queue. Some processes are ready they just need CPU to get executed and Complete their task, those processes stay in ready queue. In multithreading several processes kept in ready queue which wait for CPU to get done with current process. In his work [1] H. Arora says that when any process wait for any I/O device or any other system interrupt the operating system get CPU back from that process and assigned other process.

In old uniprogramming environment one process run at a time and if that process has to wait for any reason CPU remain idle scheduling algorithms use to maximize the CPU utilization because CPU remaining idle is not a good approach. CPU algorithms keep track of processes and arrange them in ready queue so that they get executed when CPU finished with current working process.

In [1] Arora suggested that pipe lining concept can be used in CPU scheduling. As CPU fetch, decode and execute the

process which take time while CPU is fetching the process the decoding and executing cores are free after implementing pipe lining when CPU will be busy in fetching the process the decoding and execution or old processes can take place. This can make more efficient CPU utilization.

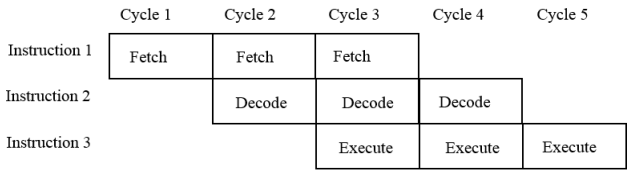

Fig. 1. Pipe Lining in CPU

When some processes that invoke at same interval or may be at different time interval and they are in ready queue. Where these processes are waiting for CPU to get executed. Here an issue arise which process have to execute first and which is to second, for this purpose we implement scheduling. In [6] Abraham Silberschatz concluded that scheduling is most important job of an operating System our Operating system have two types of schedulers which are short-term scheduler (CPU scheduler) and long-term scheduler (Job scheduler). In [8], [2] A. Silberschatz and M. Sindhu say that short-term scheduler decide which processes should executed next and assign CPU and long-term scheduler decide which job is ready enough to brought in to ready queue. Short-term scheduler is very fast while long-term scheduler is slow. When Short term scheduler bring ready processes to CPU dispatcher invoke here and do context switching which includes saving current process and starting the new process. We do scheduling by different CPU scheduling algorithms. Algorithms are finite set of rules which are used to get desire output. Different scheduling algorithm has different properties. Mainly we have four types of CPU scheduling algorithms which are First come first serve, shortest job first, priority and Round Robin. Optimization criteria for these CPU scheduling algorithms includes Maximize CPU utilization, Maximize throughput, minimize starting time, wait time and turnaround time. On the basis of properties every algorithm produces different result.

Now we have to figure out which algorithm is more efficient for CPU scheduling, for this purpose we have to compare the below mention algorithms with respect to their Response, turnaround and wait time. In [8]-[11], A. Silberschatz, S. A. Tanenbaum, and P.Kokkinsis says that algorithms are primitive and non-preemptive, Non-primitive algorithms are such algorithms which once acquire CPU they only release it when they complete their job while in primitive scheduling CPU switched to better option like when a process of low priority which high burst time is using the CPU and a process arrives which high priority short term scheduler switch the CPU to high priority process. In this case context switching time can be high. When we talk about burst time it's the total execution time of any process, and the arrival time is the time interval when a process arrive in the ready queue. In [8], [2], [13] A. Silberschatz, M. Sindhu and S. Shah explores that In primitive CPU scheduling dispatcher is use for the switching processes between CPU, it can save the current state or progress of Process and load the current progress of next process from memory it must be very fast because it can reduce context switching time.

II. RELATED WORK

CPU scheduling in important task in modern multiprogramming system. For this purpose many algorithm can be used. Some of them are discussed above. In [20] C.L. LIU says those researchers are continuously trying to increase the efficiency of these algorithms, by comparing them and adding new techniques, and combining two algorithms also. In [18] Neetu Goel use diagrams description of different CPU scheduling algorithm. In their paper they present different state diagrams to make comparison between different algorithms using for single processor system and drive the result for best CPU scheduling algorithm by different examples. In [19] Y.A. Adekunle as compare different algorithms in the basis of six parameters which are (wait time, response time, throughput, fairness, CPU utilization, starvation, preemption, and predictability) he concluded his work as there is a lot of need is to be worked on these algorithms.

In [4] N Kumar has proposed a new algorithm, which uses SJF and priority algorithm's properties with round robin CPU scheduling algorithm. Priority is intended on the foundation of SJF and quantum time. It keep the advantages and the properties of Simple Round Robin and decreases starvation and also mixes the benefit of priority scheduling algorithms. In his paper the proposed algorithm assign new priorities to the processes using SJF and quantum time. In [3] author mixed the proprieties of priority algorithm with round robin, he proposed the new idea as the reassigning the priorities according to remaining execution time of processes in round two, in round one processes are executed in assigned priorities.

In [12] P K Mittal analyzed the traditional Round Robin which result in longest wait time and a lot of context switching due to static quantum time. In his paper he proposed a new algorithm EDRR (Efficient Dynamic Round Robin) in which he uses SJF instead of FCFS and uses dynamic time quantum rather than static one, he has generated new quantum time using mean and median of burst time as different formulas, he concluded this with different testing results that is EDRR is more efficient.

III. COMPARISON PARAMETERS

Comparison of different algorithms can be done on different criteria which have different units. After comparison we are able to define the properties of different CPU scheduling algorithms, A CPU scheduling algorithm's efficiency will be depend on its average response time, Average wait time and average turnaround time.

A) Response Time

Response time is the time period taken by an algorithm to response for the very first time to a process. When a process arrives in the ready queue it sort according to different criteria

which can be based on different scheduling algorithms. Then the processes have to wait form the CPU allocation so that they can complete their task. In [6] Abraham Silberschatz defined an Equation for Response Time.

$$Response\ Time = 1^{st}\ start\ Time - arrival\ Time \quad (1)$$

B) Wait Time

The time period which is tagged as wait time of a process is total time which a process spend in ready queue waiting for CPU it also include response Time, wait time of a process depend on the scheduling algorithms its mostly high in First come First serve algorithm as the high aged processes have to execute as the arrive earlier and the short age process which arrive after them have to wait. In [6] Abraham Silberschatz defined an equation for wait time.

$$Wait\ Time = (1st\ start\ Time - arrival\ Time) + (2nd\ Start\ Time - First\ end\ Time) + \ldots + (nth\ Start\ Time - (n\text{-}1)th\ end\ Time) \quad (2)$$

In [17] Kumar Saroj has suggested an equation for average wait time.

$$Average\ Waiting\ Time\ (AWT) = \{\Sigma WT_j\}/n$$

Here, WT_j is the waiting time of j^{th} process and n is over-all number of processes.

C) Turnaround Time

Turnaround time is the total age of a Process which it spend in the ready queue and in CPU, simple turnaround time is Sum of wait time and execution time of a process. Execution time is considered as the time when CPU is allocated to that particular process and the process is performing its task. In [6] Abraham Silberschatz defined an equation for computing turnaround time.

$$Turnaround\ Time = Finish\ Time - arrival\ Time \quad (3)$$

In [17] Kumar Saroj has suggested an equation for average turnaround time.

$$Average\ Turnaround\ Time\ (ATAT) = \{\Sigma TAT_j\}/n$$

Here, TAT_j is the turnaround time of j^{th} process and n is over-all number of processes.

D) Efficiency

In general efficiency is when an algorithm performs maximum utilization of CPU. In [6] Abraham Silberschatz says that CPU utilization can be defined in parentage 0 to 100, in real system mostly system are utilizing CPU up to 40% and highly loaded system can utilize up to 90%. Efficiency can be drawn as the main goal which is to increase the throughput and decrease the wait Time of process also decrease the response time.

IV. METHODOLOGY

In modern system there are multicore processors which are executing multi processes at a time using context switching. These processes are scheduled by different CPU scheduling algorithms, which are further discussed below, we are making comparisons between these algorithms with respect to their average wait time, response time, and turnaround time. While comparing the result of different algorithms we used bar chart to clearly scan out the best CPU scheduling algorithm.

A) First-Come _ First-served Algorithm for Scheduling

This is a simple algorithm use for scheduling CPU Processes. In [1] Himanshi Arora say the FCFS is the algorithm in which processes get executed according to their arrival in ready queue. Ready queue is a place where those process stay, that are ready for execution and waiting for CPU allocation, in FCFS scheduling the concept of queue is used which is FIFO (First in first out) which means sequence of execution based on arrival time of processes. When a process is created and it brought up in ready queue by long term scheduler, if CPU is idle the Short Term scheduler directly allocated it to that process in spite of its long burst Time, In this case short age processes have to wait for if a higher age processed is currently executing which result in higher wait time according to [2] by M. Sindhu. As you know our main focus is to reduce the wait time and increase through Put. When we use same processes with ascending order with respect to burst time it produce different result. Waiting for larger burst time process to finish cause convoy affect which the process of short burst time have to wait. We take an example for FCFS CPU scheduling algorithm.

Table I: FCFS Input Parameters

Process Name	Arrival Time	Burst Time
P_1	0.0	20
P_2	1.0	6
P_3	2.0	3

In Table I there are three processes which arrive in ready queue as shown FCFS is non primitive algorithms. So in this case once CPU is allocated to a process it have to complete its task before releasing the CPU.

Steps of execution of FCFS

1. P_1 arrives first and the short term scheduler allocate CPU to P_1,
2. P_2 arrives in ready queue after P_1 and P_1 have to wait for P_2 to complete its execution and free the CPU.
3. Then P_3 arrives and it also has to wait for CPU to get wakened.
4. When P_1 releases the CPU, P_2 start execution and after this P3 start its execution and complete its task.

Gant chart

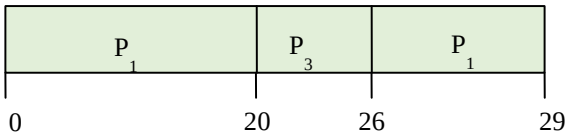

Fig. 2. FCFS Processes Sequence (1st case)

Calculation

Wait Time = start time – arrival Time

Total Wait Time= (0-0)+(20-1)+(26-2)

Total Wait Time = 43

Average Wait Time= 14.33

Total Turnaround Time= Finish Time – arrival Time

Total Turnaround Time= (20-0)+(26-1)+(29-2)

Total Turnaround Time =72

Average Turnaround Time= 24

Suppose these processes arrive in reverse order P_3, P_2, P_1. Then calculation result had notified change in wait time, and turnaround Time also. Let's take a look in this matter.

Gant chart

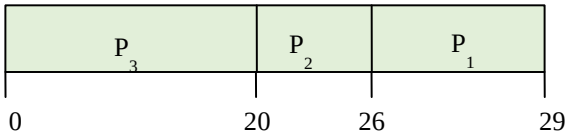

Fig. 3. FCFS Processes Sequence (2nd case)

Calculation

Wait Time = start time – arrival Time

Total Wait Time= (0-0)+(3-1)+(9-2)

Total Wait Time = 10

Average Wait Time= 3.33

Total Turnaround Time= Finish Time – arrival Time

Total Turnaround Time= (3-0)+(9-1)+(29-2)

Total Turnaround Time =38

Average Turnaround Time= 12.66

Analyzing above Cases:

In Fig. 1 we can see that there is a lot of difference in wait time and turnaround time just because we are executing the process in different order. This changing order will increase the optimization criteria which is it is decreasing the wait time and turnaround time.

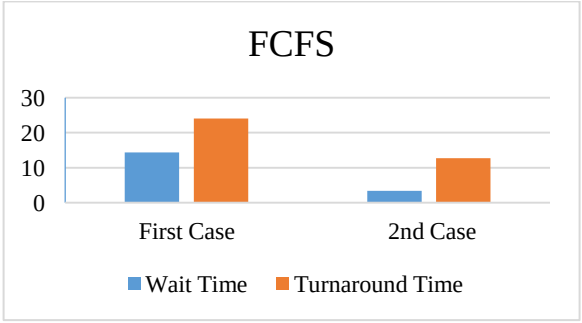

Fig. 4. FCFS 1st case vs 2nd case

B) Priority Algorithm for Scheduling

In [14], [15], [16] R. Matarnesh, J.Lakma and Md. A. F. AlHusainy says that Priority is the importance of any process for example many processes are running and a new process invoke with the job to display an message about system error which if delayed can cause data lost it have the highest priority now short term scheduler have to pick this process and assign CPU and move the current working process to Ready Queue if not completed. Priority based algorithm is quite agreeable as it better than First come first served algorithm as operating system attached a priority bit with each process on the basis of which process get executed. This priority bit can be decided on the basis of many parameters (e.g., load, resources, and importance). Priority bit is an integer value which attached to a process if this integer value is low then priority is high if the bit value is high the priority is low. Priority algorithm can be primitive like if a process with high priority arrives the short term scheduler switched the CPU between them or non-primitive in which either a process with high priority arrives the current process only release CPU only if it had completed its job. Let's take a look by an example

Table II: Priority Input Parameters

Process Name	Arrival Time	Burst Time	Priority
P_1	5	10	1
P_2	0	6	5
P_3	3	8	4

In Table II there are three processes which arrive in ready queue at different time interval with different priority and have different burst times.

1st case: Steps of execution of priority algorithms

1. P_2 arrives at 0 as it has the lowest priority but in this context it is the only process which arrive at this time interval so short term scheduler allocate CPU to P_2.
2. As this is non-primitive case so once the CPU is allocated it only releases when job is done so as P_2 has finished its job P_1 and P_3 arrives in ready queue.
3. Then short term scheduler decides on the basis of priority, P_1 have highest priority then P_3. So CPU is allocating to P_1 and after its job completion CPU is allocate to P_3.

Gant Chart

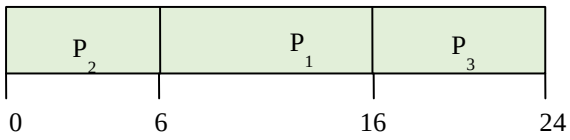

Fig. 5. Priority Processes sequence (1st case)

Calculation

Wait Time = start time – arrival Time

Total Wait Time= (6-5)+(0-0)+(16-3)

Total Wait Time = 14

Average Wait Time= 4.66

Total Turnaround Time= Finish Time – arrival Time

Total Turnaround Time= (16-5)+(6-0)+(24-3)

Total Turnaround Time =38

Average Turnaround Time= 12.66

2nd Case: Steps of execution of priority algorithms

1. Same like above mention case CPU is allocated to P_2 as it arrives at 0.
2. After starting execution short term scheduler keep looking for the highest priority process in ready queue then the current executing process because of primitive case.
3. When P_3 with the priority higher than the current executing process arrive short term scheduler allocate the CPU to P_3. as P_2 has not finish its job yet, so it have to save the current progress and again sent to ready queue this task is done by dispatcher. Which just invoke and perform the task and disposed.
4. When P_1 arrives CPU is allocated to P_1 which is of high priority and P_3's progress is saved and then sends in ready queue for wait. Now P_1 complete its job and free CPU
5. Now dispatcher Load the P_3 And short term scheduler allocate CPU to P_3 when it complete its Job CPU is allocated to P_2.

Gant Chart

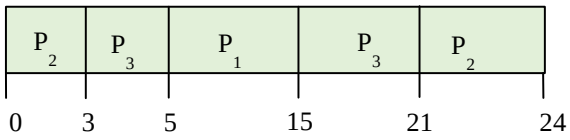

Fig. 6. Priority processes sequence (2st case)

Calculation

Wait Time = start time – arrival Time

Total Wait Time= (5-5)+(0-0)+(21-3)+(3-3)+(15-5)

Total Wait Time = 28

Average Wait Time= 9.33

Total Turnaround Time= Finish Time – arrival Time

Total Turnaround Time= (15-5)+(24-0)+(21-3)

Total Turnaround Time =52

Average Turnaround Time= 17.33

Analyzing above Cases:

In above mention cases of priority scheduling algorithm same example is solved by primitive and non-primitive method and there is a clear difference let's observe it through bar chart showed in Fig. 7.

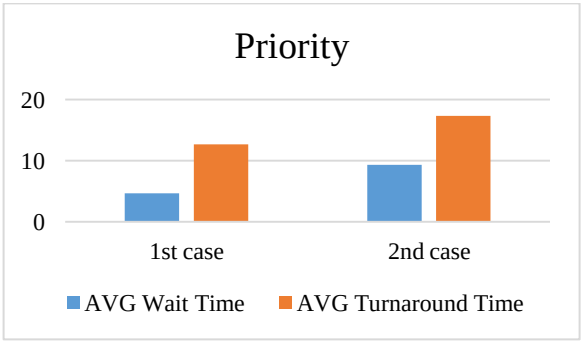

Fig. 7. Priority 1st case Vs 2nd case

As we know over main goal is to minimize the average wait time and average turnaround time, so we can clearly see that through primitive way wait time is increasing as wait time increases the turnaround time also increases.

In Priority scheduling algorithm the processes of low priority get ignored by CPU as the processes of high priority bombarded and CPU remain busy with them [2] so process suffer from starvation. This problem can be solved by increasing the Priority bit of process by fix interval of time and this method is called ageing. Through this every process got the turn to execute either if it had a lowest priority at the arrival.

C) Shortest-Job First Algorithm for scheduling

In SJF procedure processes are places in ready queue. Where short term scheduler assign CPU to them and the get executed on the base of their age or execution time called the burst time. In [2] M. Sindhu says that in SJF the process which have less execution time got executed first. Short term scheduler places the processes with the smallest burst time in head of the queue and lengthiest burst time in tail of the queue. In [16] Md. A. F. AlHusainy declares that SJF CPU scheduling algorithm can be primitive or non-primitive, in primitive case when best job that has the short burst time get in ready queue dispatcher switched the CPU to the new best coming job, and after its completion the next choice can be made by short term scheduler using above criteria. Let's take look through an example.

Table III: SJF Input Parameters

Process Name	Arrival Time	Burst Time
P_1	5	10
P_2	0	9
P_3	2	6
P_4	3	4

In Table III there are four processes (P_1, P_2, P_3, and P_4) with different burst times and different arrival times available.

1st case: steps of execution shortest job first

1. P_2 arrives at 0 in this context and short term scheduler allocates CPU to it as there is no other best option.
2. As this is a non-primitive case so CPU get free only when Job is Done so as soon as P_2 finished other all processes are arrived in ready queue.
3. Now short term scheduler decide which Job got least burst time, and in this example P_4 will get the CPU.
4. After P_4, P_3 and then P_1 will got CPU to complete its job.

Gant Chart

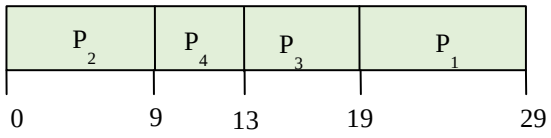

Fig. 8. SJF processes sequence (1st case)

Calculation

Wait Time = start time – arrival Time

Total Wait Time= (19-5)+(0-0)+(13-2)+(9-3)

Total Wait Time = 31

Average Wait Time= 7.75

Total Turnaround Time= Finish Time – arrival Time

Total Turnaround Time= (29-5)+(9-0)+(19-2)+(13-3)

Total Turnaround Time =60

Average Turnaround Time= 15

2nd case: steps for execution Shortest job first

1. In this context P_2 arrives at 0 so CPU is allocated to it,
2. At 2 P_3 arrives and the remaining burst time of P_2 is greater than P_3 so Dispatcher do context switching, and CPU is now assigned to P_3.
3. Then At 3 P_4 arrives same as above CPU switched to it, then P_1 arrives but it have larger burst time then P_4 so CPU remain allocated to P_4.
4. Once P_4 Complete its Job short term scheduler allocated CPU to shortest Job which is P_2 and then P_1.

Gant chart

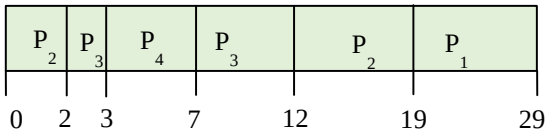

Fig. 9. Priority processes sequence (2st case)

Calculation

Wait Time = start time – arrival Time

Total Wait Time= (19-5)+[(0-0)+(12-2)]+[(2-2)+(7-3)]+(3-3)

Total Wait Time = 28

Average Wait Time= 7

Total Turnaround Time= Finish Time – arrival Time

Total Turnaround Time= (29-5)+(19-0)+(12-2)+(7-3)

Total Turnaround Time =57

Average Turnaround Time= 14.25

Analyzing above Cases:

In above mention cases of shortest job first CPU scheduling algorithm which are primitive and non-primitive let's find out which case decreases the waits time or turnaround time through bar chart.

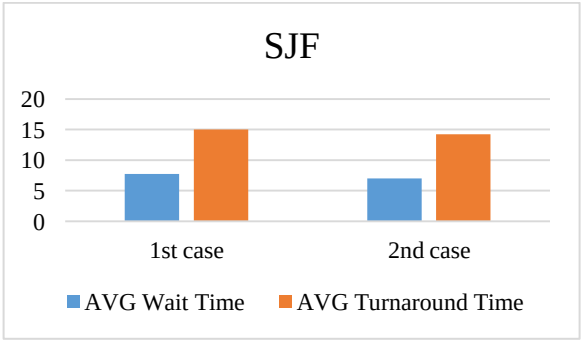

Fig. 10. Priority 1st case Vs 2st case

As we can see that in shortest job first primitive scheduling is more useful as it produce low wait time. In [18] Goel, N concluded that shortest job first algorithm is not well and efficient for processes with long burst time because in SJF short job executes first and this can cause long waiting time for these processes.

D) Round-Robin Algorithm for Scheduling

In R_R (Round Robin) procedure a fixed interval of time in given to processes to get executed by CPU that time is called quantum time it's mostly 10-100ms. In [19] Adekunle, Y.A. says that If quantum time is low then context switching should be high, and if quantum time is very large it can cause increase in response time. When that fix unit time completed CPU switched to other process further more using R_R

operating system can use FCFS as primary approach as well as SJF and Priority. In [6] Abraham Silberschatz says that A basic advantage of RR is its fairness because every process get equal amount of CPU time, like if there are n processes in the ready queue and the time quantum is q, then each process gets 1/n of the CPU time in portions of at most q time units at once. Every process can wait for (n-1)q time units not more than this.

In RR scheduling algorithm Short term scheduler use different sub algorithms to select a process from ready queue, it can be FCFS, SJF or priority, and quantum time can be fixed or dynamic let's take a look by example.

Table IV: RR Input Parameters

Process Name	Arrival Time	Burst Time	Priority
P_1	2	5	3
P_2	0	10	1
P_3	4	15	5
P_4	1	20	2
P_5	3	25	4

In Table IV there are five processes (p_1, p_2, p_3, p_4, p_5) with different arrival time, burst time and priority. And the quantum Time is 10ms. Let's see the sequence of process execution by Gant charts

RR with first come first serve

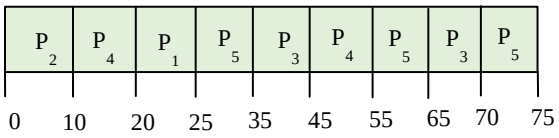

Fig. 11. RR (FCFS) processes Sequence

Calculation

Wait Time = start time – arrival Time

Total Wait Time= (20-2)+(0-0)+[(35-4)+(65-45)]+[(10-1)+(45-20)]+[(25-3)+(55-35)+(70-45)]

Total Wait Time = 180

Average Wait Time= 36

Total Turnaround Time= Finish Time – arrival Time

Total Turnaround Time= (25-2)+(10-0)+(70-4)+(55-1)+(75-3)

Total Turnaround Time =225

Average Turnaround Time= 45

RR with Shortest Job First

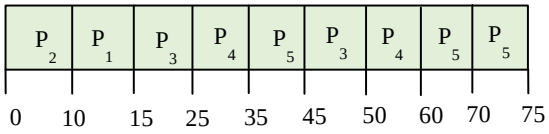

Fig. 12. RR (SJF) processes Sequence

Calculation

Wait Time = start time – arrival Time

Total Wait Time= (10-2)+(0-0)+[(15-4)+(45-25)]+[(25-1)+(50-35)]+[(35-3)+(60-45)]

Total Wait Time = 125

Average Wait Time= 25

Total Turnaround Time= Finish Time – arrival Time

Total Turnaround Time= (15-2)+(10-0)+(50-4)+(60-1)

Total Turnaround Time =205

Average Turnaround Time= 41

RR with Priority scheduling

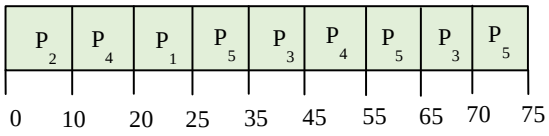

Fig. 13. RR (Priority) processes Sequence

Calculation

Wait Time = start time – arrival Time

Total Wait Time= (20-2)+(0-0)+[(35-4)+(65-45)]+[(10-1)+(45-20)]+[(25-3)+(55-35)+(70-65)]

Total Wait Time = 150

Average Wait Time= 30

Total Turnaround Time= Finish Time – arrival Time

Total Turnaround Time= (25-2)+(10-0)+(70-4)+(55-1)+(75-3)

Total Turnaround Time =225

Average Turnaround Time= 45

Analyzing above Cases:

In above mention example of Round Robin scheduling algorithm with FCFS, SJF and priority we can see the variation in wait time and turnaround time so by analyzing this we can decide which approach is best for round robin CPU scheduling algorithm,

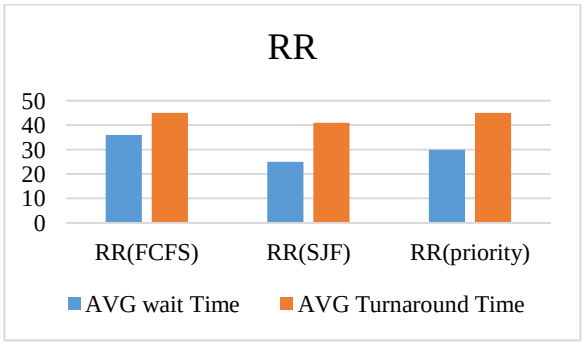

Fig. 14. Comparison of RR (FCFS), RR (SJF) and RR (priority)

Our main goal behind the CPU scheduling is to reduce the average wait time and average turnaround time and increase throughput, by above graph we can analyze the Round Robin with Shortest Job First approach is more useful than FCFS and priority.

In [4] N Kumar explores that in R_R SJF Operating system select the process which has shortest job and allocate CPU for one unit time called quantum time after accomplishment of single time quantum, match the time quantum with the remaining processes burst time if remaining burst time is less than or equal to time quantum assigns the same process again. Else repeat.

As we know simple R_R algorithm as several drawbacks (e.g., low throughput, high wait time) which can be minimized using different approaches. In [3] Rajput, I.S. suggested that operating system can use old fashion Round Robin with Priority Bit Concept which Help it to increase throughput, as unit time can be allocated to processes on the base of priority bit attached with processes. After one complete cycle processors are arranged in increasing order or their remaining CPU burst time in the ready queue. New priorities are assigned according to the remaining CPU bursts of processes. The process with shortest remaining CPU burst is assigned with highest priority. With this concept processes got new priority after every cycle then no need of ageing.

E) Experimental Setup

In Table V the average Wait time, response Time and the turnaround time is computed using self-created simulation in Object Oriented Programming Language JAVA. In which we Use a CPU for execution of processes, and number of processes can be used for each scheduling are dynamic but we mostly uses five processes in each case as a sample size. The processes burst time, arrival time and priorities was already defined before submission for execution and calculation of wait time, response time, and turnaround time. Quantum time is also already defined.

V. SIMULATION RESULTS

The environment in which these simulation take place is a single CPU system, and burst times and priorities are fixed and already known, quantum time used in Round Robin algorithm is static and in milliseconds. While performing number of examples using different algorithms which are (FCFS, SJF, Priority, RR (FCFS), RR (SJF), RR (Priority). These results are coated below where Table V is result of response time, Table VI is result of wait time, Table VII is turnaround time. E stands For Example.

Table V: CPU scheduling Algorithms Average Response Time

Examples	E:1	E:2	E:3	E:4	E:5	E:6	E:7	E:8	E:9	E:10	AVG
FCFS	6.4	10	12.6	12.8	4.4	11.6	3.2	9	3	10.4	8.34
Priority	8.6	9.2	13.4	12.4	8.8	10	3.2	9.6	4	10	8.92
SJF	4.8	8.2	9.6	11	4	8	2.4	8.8	3	9	6.88
RR (FCFS)	3.8	5.6	5.8	6.4	3.4	7.4	3.2	5.6	2.8	3.6	4.76
RR(Priority)	5	7.5	5.8	6.4	4.6	7.4	3.2	5.6	3.2	3.2	5.19
RR(SJF)	4.2	6.6	5.8	6.4	3.4	7	2.6	5.6	2.8	3.2	4.76

In Table V there are ten different examples including 5 processes each with different burst times, different arrival time, and for priority scheduling different priorities, used, in Table V we have average response times for each example using different algorithms which are FCFS, SJF, Priority, and Round Robin (FCFS, SJF, and Priority).

In Fig. 15, we can analyze the response time of these examples and pull a result which algorithm is more useful in decreasing response time. As we now every algorithm has different properties due to which we have verities of results but round robin algorithm has produced low response time, which means that every process got CPU for the first time in less Time.

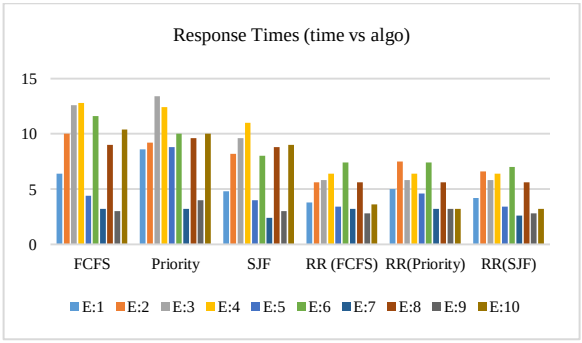

Fig. 15. Comparison of Response time

In Table V we have ten different examples which are solved by 6 different algorithms. In Table V the values are of average wait time for each example using different CPU scheduling algorithm. Wait time varies due to variation in the properties of CPU scheduling algorithms.

Table VI: CPU scheduling Algorithms Average Wait Time

Examples->	E:1	E:2	E:3	E:4	E:5	E:6	E:7	E:8	E:9	E:10	AVG
FCFS	6.5	10	12.6	12.8	4.4	11.6	3.2	9	3	10.4	8.35
Priority	8.6	9.2	13.4	12.4	8.8	10	3.2	9.6	4	10	8.92
SJF	4.8	8.2	9.6	11	4	8	2.4	8.8	3	9	6.88
RR (FCFS)	10.6	12.2	16.6	20.6	5.4	16.2	3.2	15	4	13.4	11.72
RR(Priority)	10.6	10	15.6	19.4	7.8	15.6	3.2	15.6	6	13.4	11.72
RR(SJF)	9.4	8	12.6	17	5	12.6	2.6	14.8	4	11.8	9.78

In Fig. 16, by Table VI we can make a graphical analysis of wait time for each example in each algorithm, we can see that wait time is decreases in shortest job first CPU scheduling algorithm, because in round robin wait time increases due to long burst times,

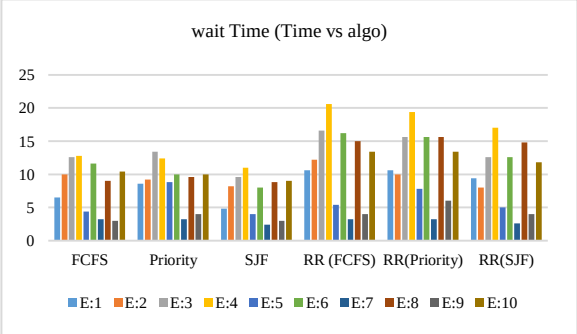

Fig. 16. Comparison of Wait time

In Table VI there are ten different examples denoted as E:1-E:10 which are solved by different six CPU scheduling algorithms, the values are of average turnaround time, which are computed using simulation. Turnaround time is actually sum for burst time and wait time as we have to minimize the wait time so turnaround time automatically decreases.

Table VII: CPU scheduling Algorithms Average Turnaround Time

Examples	E:1	E:2	E:3	E:4	E:5	E:6	E:7	E:8	E:9	E:10	AVG
FCFS	11.2	15.6	20.6	21	10	18	6.6	16.4	7.8	19	14.62
Priority	13.4	14.8	21.4	20.6	14.4	16.4	6.2	17	8.8	18.6	15.16
SJF	9.6	13.8	17.6	19.2	9.6	14.4	5.4	16.2	7.8	17.6	13.12
RR (FCFS)	15.4	17.8	22.6	28.8	11	22.6	6.2	22.4	8.8	22	17.76
RR(Priority)	15.4	15.6	23.6	29.6	13.4	22	6.2	23	10.8	22	18.16
RR(SJF)	14.2	13.6	20.6	25.2	10.6	19	5.6	22.4	8.8	20.4	16.04

By Table VII we have drawn the graph in Fig. 17 through which we can analyze which algorithm is best for decreasing the turnaround time, in below mention chart we can see that by using SJF CPU scheduling algorithm, average turnaround time of every example is less than 20ms, but in other algorithms turnaround time is crossing this limit.

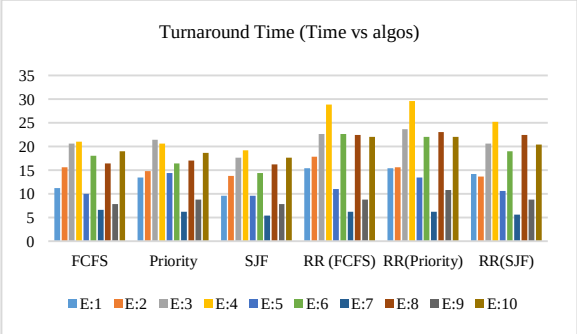

Fig. 17. Comparison of Turnaround time

By using above examples we have to calculate the averages of these results which can be further use for the critical analysis. Table 7 contain the average results of response time, wait time and turnaround time which is produced from different examples using six different CPU scheduling algorithms.

Table VIII: Averages of Response, wait and turnaround Time

	AVG Response Time	AVG Wait Time	AVG Turnaround Time
FCFS	8.34	8.35	14.62
Priority	8.92	8.92	15.16
SJF	6.88	6.88	13.12
RR (FCFS)	4.76	11.72	17.76
RR(Priority)	5.19	11.72	18.16
RR(SJF)	4.76	9.78	16.04

In Fig. 18. There are six different algorithms which are first come first serve, shortest job first, priority, Round robin with FCFS, Round Robin with SJF, Round Robin with Priority. On the horizontal axis there are algorithms and on vertical axis there is time. Which is computed from a lot of examples, mention in Table V, Table VI, and Table VII.

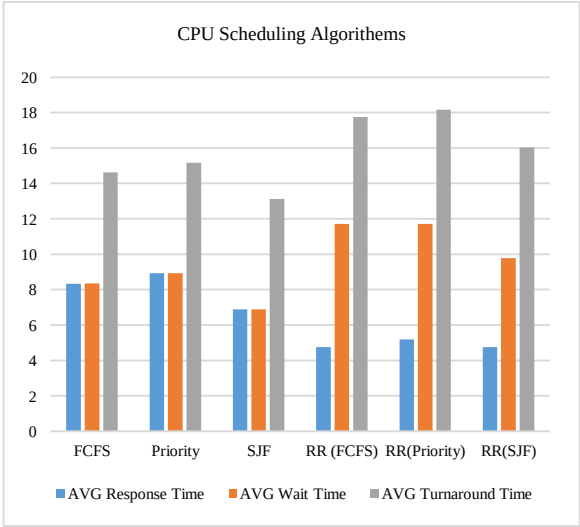

Fig. 18. Comparison of Average Results

VI. CONCLUSION

CPU scheduling is done by various algorithms that possibly the one can be the combination of two different algorithms. We have done the comparison of these different CPU scheduling algorithms which are FCFS, SJF, Priority, Round Robin (FCFS), Round Robin (SJF) and Round Robin (priority). This comparative analysis is done on the basis of different criteria, which include average wait time, average response time and average turnaround time. We have analyzed that which algorithm minimized the wait time, response time and turnaround time.

Our analysis criteria include solution of 60 different example with all these six above mention CPU scheduling algorithms, the results are then critically analyzed through bar chart which show clearly which algorithm is best according to our context. As we know our main goal is to increase the throughput and decrease the wait time and response time. So after performing different analysis we come to conclusion that between these algorithms (FCFS, SJF, priority, RR (FCFS), RR (SJF) and RR (priority)). Shortest job first CPU scheduling algorithm is best due to decreasing in average wait

time and average response time but as we know technology is continuously improving day by day so these algorithm are also improving by combining different algorithm which can be more efficient.

Future Work

As we can see that due to technology advancement there are a lot of new ideas are generating, CPU scheduling can be made more efficient by different and unique algorithms which may be the combination of two or more stand-alone CPU scheduling algorithms. Which produce more effective result for CPU scheduling.

REFERENCES

[1]. H. Arora, D. Arora, B. Goel and P. Jain, "An Improved CPU Scheduling Algorithm", International Journal of Applied Information Systems, Vol. 6, No. 6, pp. 7-9, 2013.

[2]. Sindhu, M., Rajkamal, R. and Vigneshwaran, P., "An optimum multilevel CPU scheduling algorithm", Advances in Computer Engineering (ACE), 2010 International Conference, (pp. 90-94), 2010, June.

[3]. Rajput, I.S. and Gupta, D., A priority based round robin CPU scheduling algorithm for real time systems. International Journal of Innovations in Engineering and Technology, 1(3), pp.1-11,2012.

[4]. N Kumar, Nirvikar, Performance Improvement Using CPU Scheduling Algorithm-SRT, International Journal of Emerging Trends & Technology in Computer Science (IJETTCS) Vol. 2, Issue 2, March – April 2013.

[5]. N Goel, Dr. R. B. Garg, Performance Analysis of CPU Scheduling Algorithms with Novel OMDRRS Algorithm. (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 7, No. 1, 2016

[6]. Abraham Silberschatz," CPU scheduling," in Operating System Concepts, ix ed. Hoboken, NJ: Wiley, 2013.

[7]. Andrew s. Tanenbaum," Introduction" in Modern Operating Systems, iii ed. Harlow:Pearson Education, 2015

[8]. Silberschatz, P. Galvin, G. Gagne, "Applied Operating System Concepts, "First edition john Wiley Sons Inc., 2000.

[9]. S. A. Tanenbaum, A. Woodhull, "Operating System, Design," Prentice Hall; 3rd editions, 2006.

[10]. P.Kokkinsis, "A Software Tool for process Scheduling," report, 2007.

[11]. Congnizant Handout, "Fundamentals of Computer Technology," Version:FCT/Handout/0307/7.1, 2007.

[12]. Mittal, P.K., An Efficient Dynamic Round Robin CPU Scheduling Algorithm. IJARCSSE Journal, 4(5), 2014.

[13]. S. Shah, A. Mahmood, A. Oxley,"Hybrid Scheduling and Dual Queue Scheduling,"Computer Science and Information Technology, 2009.

[14]. R. Matarnesh, "Self Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running processes," American Journal of Applied Science, 2009.

[15]. J.Lakma, "Fortifying the Operating System CPU Scheduler," A project report submitted to csed of Makerere Uni., 2005.

[16]. Md. A. F. AlHusainy,"Best -Job-First CPU Scheduling Algorithm," Information Technology Journal, 2007.

[17]. KumarSaroj, S., Sharma, A.K. and Chauhan, S.K., April. A novel CPU scheduling with variable time quantum based on mean difference of burst time, In Computing, Communication and Automation (ICCCA), 2016 International Conference on (pp. 1342-1347). IEEE. 2016

[18]. Goel, N. and Garg, R.B., A comparative study of CPU scheduling algorithms. arXiv preprint arXiv:1307.4165. 2013.

[19]. 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, Int. J. Innov. Sci. Res, 12(1), pp.180-185, 2014.

[20]. C.L. LIU, JAMES W. LAYLAND, Scheduling Algorithm for Multiprogramming in a Hard-Real-Time Environment, Journal of the ACM, Vol. 20, Issue 1, PP. 46 – 61, Jan. 1973.