International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 08 | Aug 2021 www.irjet.net p-ISSN: 2395-0072

A New Proposal of Smart Time Quantum for Round Robin Algorithm and Comparison with Existing Round Robin Variants

Edula Vinay Kumar Reddy[1], Kotha v v s Aakash[2]

[1]*B.Tech Student, Computer Science Engineering, VIT University, Vellore, Tamil Nadu, India*
[2]*B.Tech Student, Computer Science Engineering, VIT University, Vellore, Tamil Nadu, India*

Abstract – *CPU Scheduling is the prime concept of Operating Systems. A scheduling algorithm solves the difficulty of deciding which of the outstanding processes is to be allocated to the processor. Round Robin is one of the superior scheduling algorithms among many CPU scheduling algorithms where every process in the queue executes for a given time quantum. But logically, every distinct case has a particular Time quantum at which the execution of Round Robin algorithm will be at its peak. Thus, the Performance and CPU efficiency are primarily dependent on the value of Time quantum we provide. Since, the value of Time quantum in Traditional Round Robin is independent of the data of processes, the CPU performance may not be best in all the cases. So, in this paper a modified Round Robin algorithm is proposed, which uses the approach of Smart Time quantum to make the traditional Round Robin Algorithm more efficient. The proposed algorithm improves the efficiency of the traditional Round Robin algorithm where the smart Time quantum is generated by taking the burst times of the processes into consideration. As a consequence, waiting time, turn-around time, and the number of context switches are reduced effectively. This paper also presents a detailed comparative study of different Existing Round Robin variants with our proposed algorithm.*

Key Words: Round Robin, Arrival Time, Burst Time, Waiting Time, Turn Around Time, Response Time, Completion Time, Smart Time Quantum, etc.

1.INTRODUCTION

As a matter of fact, in a Single-Processor system, only one process can run at a particular point of time, and any other processes must wait until the CPU becomes free and can be rescheduled accordingly. In order to start execution, most of the processes must wait, typically for the completion of some I/O request. So, in a Single-Processor computer system, the CPU just sits idle for completion of such I/O requests. All this waiting time is wasted; i.e., no useful work is carried out in the meantime.

So, with the introduction of Multiprogramming, we try to use this time in a productive manner. In general, numerous processes are kept in the memory at one time. When one process has to wait for some kind of I/O request (or any other), the Operating system takes the CPU away from that particular process and allocates another process to the CPU and this pattern continues. With this kind of switching the CPU among various processes, the Operating system (OS) can make the computer to work in a more efficient manner. The prime focus of Multiprogramming is to have some process running at all the times; i.e., to amplify the CPU utilization.

CPU scheduling is the core of any Multiprogramming based Operating systems. One of the major reasons for using Multiprogramming is that the Operating system itself is implemented as one or more process(es), so there must be a way for the application processes and operating system to share the CPU effectively. Another main reason is that the processes which require to perform any kind of I/O requests in the normal course of computation, generally require more time to complete than to perform the CPU instructions, Multiprogramming systems allocate the CPU to another process whenever any process invokes a I/O operation or any other kind of interrupt.

Every time the CPU becomes idle, the Operating system must pick one of the processes among the processes present in the ready queue for the execution at that point of time. This selection procedure is carried out by the Short-term scheduler (The CPU scheduler). This Scheduler picks one of the processes present in the memory that are ready to get executed, and allocates it to the CPU.

Another component involved in this entire scheduling procedure is the Dispatcher. The Dispatcher is the component that allocates the process selected by the short-term scheduler to the CPU. The functions of Dispatcher involve:

- Context switching.
- To Restart the process from where it left last time, by remembering the proper locations.

The Dispatcher module should be as fast as possible, considering that it is invoked during every Process switch. The Time consumed by the dispatcher module to end one process and begin another process is known as the Dispatch Latency. Low Dispatch Latency directly implies high utilization of the CPU, since the wastage of time in the context of Dispatch Latency is low. Typically, in a drastic case of Dispatch Latency being high, there is no meaning for Multiprogramming.

2. SCHEDULING

Scheduling refers to the technique in which the processes are assigned to run on the available CPUs, because there are typically higher number of processes willing to run at a same time than the available number of CPUs. In other way, Scheduling can also be defined as the process of deciding of which particular process should occupy which particular resource (CPU, disk, etc.). This indicates that CPU scheduling is a method which allows one of the processes to utilize the CPU, while the execution of the another process is kept on hold state (in Waiting state) due to unavailability of any resource(s) like I/O (or any other), thereby maximizing the CPU utilization. The objective of CPU scheduling is to make the system fair, fast and proficient.

CPU scheduling decisions takes place under the following instances:

1. Once a Process switches from the Running phase to the Waiting phase (For any I/O request (or) any other).

2. Once a Process switches from the Running phase to the Ready phase (When any interrupt occurs).

3. Once a Process switches from the Waiting phase to the Ready phase (After the completion of any I/O request).

4. Once a Process terminates.

Whenever Scheduling takes place only under instances 1 & 4 shown above, we say the scheduling scheme is a called as a non-Preemptive one; Else, the scheduling scheme is called as a Preemptive one.

Under the policy of non-Preemptive scheduling, once the CPU has been allocated with any of the processes, that particular process is kept in the CPU until it gets terminated (or) switches to the Waiting phase. Whereas under the policy of Preemptive scheduling, the tasks are usually assigned with respective priorities. At times it is necessary to run a certain process that has a higher priority before another process although it is already in Running phase. So, the currently running process is interrupted for some amount of time and resumed later when the high prioritized process has finished its entire Execution.

When we think of Scheduling, various questions would be raised in our mind like:

- Which process should be given Higher priority?
- If suddenly any running process gets terminated, which other process is allocated to CPU by the Operating system?
- which process is allocated to CPU by the Operating system, If suddenly any running process invokes any I/O request?
- What is the order of priority of all the processes available in ready queue?

So, all these questions can be answered based on the type of CPU scheduling algorithm we employ. Different types of Scheduling algorithms give different order of Priorities among the available processes. There are variable types of CPU scheduling algorithms available to perform Multiprogramming like First Come First Serve (FCFS), Shortest Job First (SJF), Priority Scheduling (PS), Shortest Remaining Time First (SRTF), Multilevel feedback Queue and Round Robin (RR). Performance of these different scheduling algorithms can be measured and compared among themselves using different criteria like Waiting time, Turnaround time, Throughput, etc.

2.1 Scheduling Criteria

Different CPU scheduling algorithms have completely different set of execution phases and the selection of any particular algorithm favors one particular class of processes over the another. In making a choice of which algorithm to be used in a particular situation, we must consider the various parameters of algorithms.

- **CPU utilization**: The prime focus of any algorithm is to keep the CPU busy to a maximum extent.
- **Throughput:** If the processor is busy executing processes, then useful work is being done. One of the important measures of the useful work done is the number of processes that are completely executed per time unit, which is defined as throughput.
- **Turnaround time:** At the context of a particular process, the significant criteria is how long it takes to execute that particular process completely. Precisely, the interval between the time its arrival and time of its termination is termed to be turnaround time.
- **Waiting time:** Any scheduling algorithm does not affect the amount of time during which a process executes or performs some kind of I/O request; they majorly affect the amount of time that a process spends idlily waiting in the ready queue.
- **Response time**: In a general system, turnaround time may not be the best measure. Thus, another measure i.e., the interval between the time of arrival of a process and the time at which it gets first response (allocated to CPU) is introduced, which is termed to be response time.

2.2 Scheduling Main Objectives

- Minimizing the Turnaround Time.
- Minimizing the Waiting Time.
- Maximizing Throughput.
- Maximizing CPU utilization.
- Minimizing the Response time.
- Providing fairness to all the Processes.

3. DIFFERENT VARIANTS OF ROUND ROBIN ALGORITHM'S

3.1 Traditional Round Robin Scheduling Algorithm

The Round Robin (RR) scheduling algorithm is designed especially for time-sharing systems (A time sharing system allows many users to share the computer resources parallelly.). It is updated version of FCFS Algorithm, in which pre-emption is added to switch between the processes. So, a small instance of time, called a Time quantum, is defined. The ready queue is treated as a circular queue in the case of Round Robin.

3.1.1 Implementation of Round Robin Scheduling

- We consider the Ready queue as a FIFO (First In First Out) queue of processes.
- If any new process enters, then it is added to the tail of the ready queue.
- The Dispatcher (Short-term scheduler) selects the very first process from the ready queue of processes and sets a timer to interrupt the process immediately after 1-time quantum, and dispatches the process.
- If there arises a case of the process having a CPU burst of less than 1-time quantum, the process itself will release the processor voluntarily.
- And then the scheduler picks the next process from the ready queue.
- Otherwise, In the case of CPU burst of the current process under execution being longer than 1-Time quantum, the timer will go off exactly after 1 Time quantum and it will cause an interrupt to the OS. Subsequently a context switch will be performed, and the process will be put at the tail of the Ready queue.
- Then the scheduler selects the next process available in the ready queue.
- This recursive process goes on until all the available processes gets executed completely.

3.1.2 Advantages of Round Robin CPU Scheduling Algorithm

- Round Robin Scheduling Algorithm doesn't suffer from the problem of Starvation.
- All the Processes get a fair allocation of Processor.
- Average Response time would be very less, compared to any other scheduling algorithms.

3.1.3 Disadvantages of Round Robin Scheduling Algorithm

- Very Important jobs may wait in Queue, as priority of processes doesn't matter in Round Robin.
- Setting the time quantum too short causes too many context switches and lowers the efficiency of CPU, especially if the dispatcher latency is considerably large.
- Setting the time quantum too large may lead to poor response time and, RR indirectly runs in the form of FCFS in that case.

3.2 Improved Round Robin CPU Scheduling Algorithm (IRR)

We know that, Round Robin algorithm is used for executing the processes on CPU with a small unit of time slice. IRR Scheduling Algorithm is more or less similar to RR algorithm using time slice but little different in the execution of processes on the CPU. IRR scheduling algorithm picks the first process in the Ready queue and execute it till the allocated time slice then we check the remaining burst time of the current running process. If the remaining burst time is less than the allocated time quantum then the running process is executed again till it finishes its execution. But, if the remaining burst time of the running process is more than the allocated time slice then the next process in the ready queue is executed. And we remove the currently running process from the Ready queue and put it at the end of the ready queue [1].

3.2.1 Implementation of IRR CPU Scheduling Algorithm

- Step-1: START
- Step-2: Consider a Ready queue of the Processes.
- Step-3: Repeat steps 4, 5 and 6 Till the Ready queue becomes empty.
- Step-4: Select the very first process from the ready queue and allocate it to the CPU for the given Time slice.
- Step-5: If the left-over burst time of the currently running process is less than that of the allocated time slice, then allocate CPU again to the currently running process for execution of remaining CPU burst time. After complete execution of current process, we remove it from the ready queue and go to step 3.
- Step-6: Detach the currently running process from the Ready queue and put it at the tail of the ready queue.
- Step-7: END

3.3 An Additional Improvement Round Robin CPU Scheduling Algorithm (AAIRR)

The AAIRR CPU scheduling algorithm picks the very first process that arrives in the Ready queue and allocates it to the CPU for its execution for a time period of 1-Time quantum (TQ). After allocating it for 1-time quantum, the remaining CPU burst time of the currently running process is checked. If it is observed to be less than or equal to 1-Time

quantum (TQ), then the CPU is again allocated to the currently executing process for the execution for remaining burst time of the process and after execution it removed from the ready queue, then it selects the next shortest process from the ready queue. Otherwise, the process will be added at the end of the ready queue and the next process in the ready queue will be selected and allocated to the CPU [2].

3.3.1 Implementation of AAIRR CPU Scheduling Algorithm

- Step-1: START
- Step-2: Consider a Ready queue of the Processes.
- Step-3: Pick the 1st process that arrives to the Ready queue and allocate the CPU to it for a time interval of 1-Time quantum.
- Step-4: If the left-over burst time of the currently executing process is less or equal to 1-Time quantum. We reallocate CPU again to the currently running process for the remaining CPU burst time. After completion of the execution, then we remove it from the Ready queue.
- Step-5: Otherwise, remove the currently running process from the ready queue and put it at the end of the Ready queue.
- Step-6: If the Ready Queue is non-empty, pick the next process with shortest Burst time from the Ready queue and assign the CPU to it for a time span of 1-time quantum then go to step 4.
- Step-7: If the Ready Queue is Empty, then Calculate Number of Context Switches, Average Waiting Time and Average Turnaround Time.
- Step-8: END

3.4 An Enhanced Round Robin CPU Scheduling Algorithm (Enhanced RR)

This algorithm is an improved version of IRR algorithm. ERR selects the 1st process from the ready queue and allocate it to the CPU for a time interval of 1-Time quantum. Thereafter it checks the left-over burst time of the currently running process, if it is found to be less than or equal to 1-Time quantum, the current process is again allocated to the CPU for the execution of remaining burst time of the process. After completion of the execution, this process is removed from the Ready queue. If the remaining burst time of the currently running process is greater than that of 1-Time quantum, then the process will be added at the end of the ready queue [3].

3.4.1 Implementation of Enhanced RR CPU Scheduling Algorithm

- Step-1: START
- Step-2: Consider a Ready queue of the processes.
- Step-3: Assign the CPU to the 1st process of the ready queue for a time span of 1-Time quantum.
- Step-4: In the case of the left-over Burst time of the process under execution being equal to or less than 1-Time quantum then we allocate the CPU again to the currently executing process for the execution of remaining burst time. After completion of execution detach it from the ready queue and go to step 3.
- Step-5: Pull out the currently running process from the ready queue and put it at the end of the ready queue.
- Step-6: Go to Step-3 if the ready queue is not Empty.
- Step-7: END

3.5 Round Robin Remaining Time Algorithm (RRRT)

This algorithm is improved version of Adaptive Round Robin Scheduling algorithm. It assumes that all processes arrive simultaneously in the Ready queue, then they are arranged in an ascending order according to their burst times. TQ is calculated by using the formula Σ pi / 2n. If the left-over CPU burst time of the currently executing process is less than the TQ, the CPU is again allocated to the currently running process for the execution of remaining CPU burst time. Otherwise, the process will be added to the end of the ready queue [4].

3.5.1 Implementation of RRRT CPU Scheduling Algorithm

- Step-1: START
- Step-2: Consider a Ready queue of the processes.
- Step-3: Rearrange all the process in the increasing order of their CPU burst times.
- Step-4: Calculate TQ as time quantum = Σ (BT(Pi) / 2*n)
- Step-5: Allocate the 1st Process in Ready Queue for 1-Time Quantum, if (remaining burst time < time quantum), reallocate CPU to the current running process for remaining burst time's execution, Else remove the current running process from the ready queue and put it at the end of the ready queue.
- Step-6: If Number of processes > 0, then Goto Step-5
- Step-7: Calculate Number of Context Switches, Average Waiting Time and Average Turnaround Time.
- Step-8: END

3.6 Enriched Round Robin Algorithm (Enriched RR)

This algorithm is the improved version of Adaptive Round Robin Scheduling algorithm. This algorithm assumes that all the processes arrive simultaneously in the Ready queue, then they are arranged in an ascending order according to their burst times. TQ is calculated by 0.75*(Σ pi / N). If the left-over CPU burst time of the currently executing process is

less than the TQ, the CPU is again allocated to the currently running process for the execution of remaining CPU burst time. Otherwise, the process will be added to the end of the ready queue [5].

3.6.1 Implementation of Enriched RR CPU Scheduling Algorithm

- Step-1: START
- Step-2: Consider a Ready queue of the processes.
- Step-3: Rearrange all the process in the increasing order of their CPU burst times.
- Step-4: Calculate TQ as time quantum = 0.75*(Σ pi / N).
- Step-5: Allocate the 1st Process in Ready Queue for 1-Time Quantum, if (remaining burst time < time quantum), reallocate CPU to the current running process for remaining burst time's execution, Else remove the current running process from the ready queue and put it at the end of the ready queue.
- Step-6: If Number of processes > 0, then Goto Step-5
- Step-7: Calculate Number of Context Switches, Average Waiting Time and Average Turnaround Time.
- Step-8: END

4. PROPOSED ALGORITHM

The Proposed algorithm primarily focuses on the drawbacks of Traditional Round Robin algorithm i.e., every different case has a particular Time quantum at which the execution of Round Robin algorithm will be at its best. Thus, the CPU efficiency and performance are primarily dependent on the value of Time quantum provided. As the value of Time quantum in Traditional Round Robin is independent of provided data of processes like their Burst times and Arrival times, the CPU performance may not be best in all the cases. So, in this Paper a modified Round Robin algorithm is proposed, which uses the approach of Smart Time quantum to make the traditional Round Robin Algorithm more efficient. The proposed algorithm improves the efficiency of the traditional Round Robin algorithm where the Smart Time quantum is generated by taking the Burst times of the processes into consideration. As a consequence, Waiting time, Turn-around time, and the Number of Context switches are reduced effectively.

4.1 Assumptions

While execution of the Proposed algorithm, it has been assumed that the system where the proposed algorithm is executing is a Single processor system and all the processes have equal priority. It is also assumed that the Number of processes, Burst times and Arrival times are well-known even before submitting the processes to the processor. All processes are only CPU bound.

In the proposed algorithm, the smart time quantum will be calculated based on the Burst times of all the processes using the formula:

$$STQ = int \left(\left| \frac{MAX(BT) - MIN(BT) - MEAN(BT's)}{2} \right| + 1 \right)$$

If the above STQ results as 1 unit time in any drastic case, in order to reduce the Number of Context switches due to less TQ (1 unit time), the STQ would be re- calculated from the formula (in above formula MIN(BT's) is ignored to obtain the below STQ since Min(BT's) is too small in most of the cases and hence can be neglected to get better results in the context of Number of Context switches):

$$STQ = int \left(\frac{MAX(BT) - MEAN(BT's)}{2} + 1 \right)$$

4.2 Terminologies

- **STQ**-Smart Time Quantum
- **MAX(BT)-**Maximum Burst Time among the available Processes
- **MIN(BT)**-Minimum Burst Time among the available Processes
- **MEAN(BT's)**-Mean of the Burst times of all the available Processes

4.3 Implementation of the Proposed Algorithm

- Step-1: START
- Step-2: Arrange the processes in the expanding request of CPU burst times in the ready queue.
- Step-3: Calculate the Mean of the Burst times of all the Processes present in Ready Queue.
  Mean (BT's) = (BT1+BT2+BT3+. . . +BTn)/n
- Step-4: Set the Smart time quantum (STQ) using the formula specified below:

$$STQ = int \left(\left| \frac{MAX(BT) - MIN(BT) - MEAN(BT's)}{2} \right| + 1 \right)$$

- Step-5: In case of the above formula resulting in 1 unit time; then consider and recalculate the STQ using the formula specified below:

$$STQ = int \left(\frac{MAX(BT) - MEAN(BT's)}{2} + 1 \right)$$

- Step-6: Allocate the CPU to the First process in the Ready Queue for the span of 1 Smart Time Quantum.
- Step-7: If the left-over CPU burst time of the currently executing process is less than or equivalent to that of 1 Smart time quantum, then Reallocate the CPU to same process for the execution of rest of the burst time; After the total execution, expel it from the Ready queue; Next Update the Ready queue if any new process arrives meanwhile. Otherwise expel the currently executing process from the CPU and put it at the tail of the Ready queue, once after updating the Ready queue with the newly arrived processes, for further execution.
- Step-8: If the Ready Queue is not empty, then revert back to Step-6.
- Step-9: Calculate the Average Waiting time, Average Turnaround time, and Average Response time of all processes.
- Step-10: END

Note: Initially, if we arrive with a case of 2 or more processes having same arrival time, then we implement SJF i.e., the process with least Burst Time is executed first among the conflicting processes.

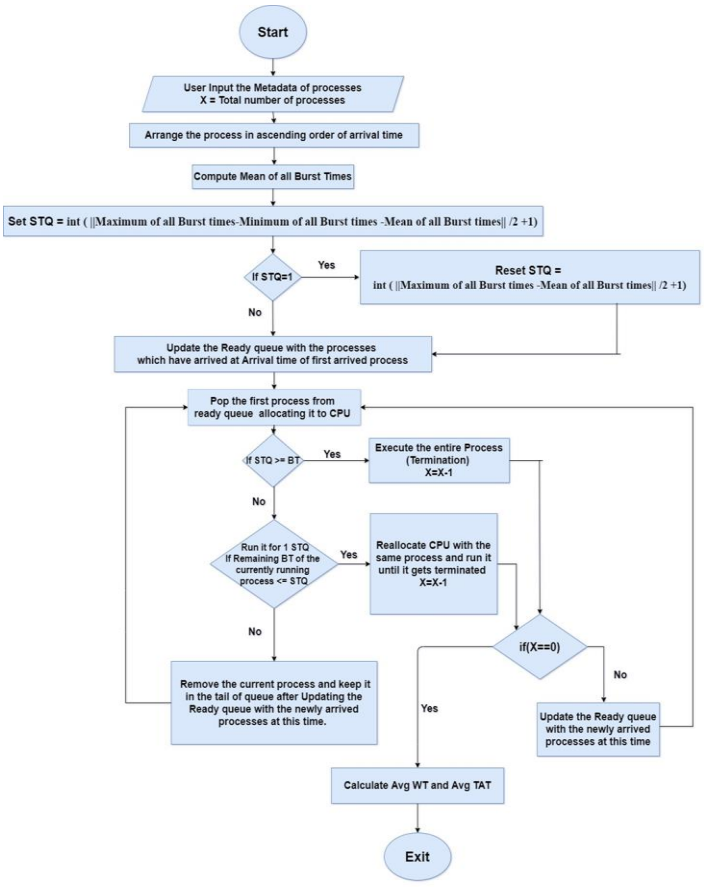

Fig -1: The Flowchart of Proposed Algorithm

5. EXPERIMENTAL ANALYSIS

5.1 Comparison of the Proposed algorithm with the Traditional Round Robin Algorithm

5.1.1 Assumptions

It has been assumed that the system where all these experiments are performed is Single processor system and all the processes have equal priority. It is also assumed that the Number of processes, Burst times and Arrival times are well-known even before submitting the processes to the processor. It is also assumed that the processes are only CPU bound.

5.1.2 Experiments Performed

Two different Cases have been taken for the performance evaluation of the Proposed algorithm.

- In Case-1, CPU burst times are taken in random order and processes arrival times are assumed to be zero.
- In Case-2, CPU burst times are taken in random order and processes arrival times are assumed to be non-zero.

5.1.3 Dataset Description

Test cases are adopted from the paper published by Manish Kumar Mishra and Dr. Faiur Rashid [6].

5.1.4 Case-1: Zero Arrival Times

In this case, CPU burst times are taken in random order and processes arrival times are assumed to be zero. A Ready queue with 5 processes P1, P2, P3, P4 and P5 has been considered as shown in table-1.

Table-1: Processes with their respective Arrival and Burst times (Case-1)

Process	Arrival Time	Burst Time
P1	0	15
P2	0	32
P3	0	10
P4	0	26
P5	0	20

5.1.5 Comparison Between Proposed Modified Round Robin & Traditional RR at Different Time Quantum's: (Case-1)

Table-2: Tabular Comparison Between Proposed Modified Round Robin & Traditional RR at Different Time Quantum's (Case-1)

Algorithm employed	Time Quantum	Avg. TAT	Avg. WT	Avg. RT
RR	1	82.00	61.40	2.00
Proposed RR	Calculated (STQ=2)	73.60	53.00	4.00
RR	2	82.40	61.80	4.00
RR	3	82.00	61.40	6.00
RR	4	81.20	60.60	8.00
RR	5	85.80	65.20	10.00

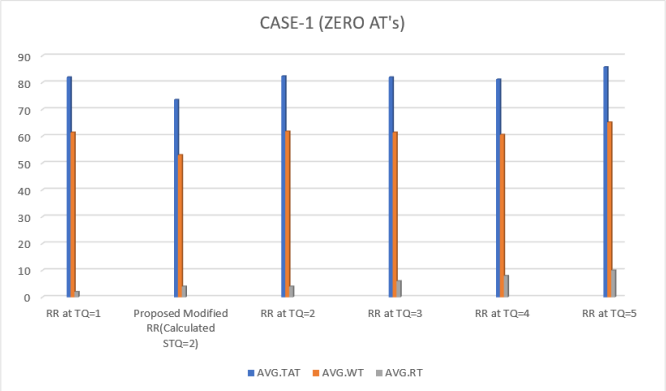

Chart-1: Graphical Comparison Between Proposed Modified Round Robin & Traditional RR at Different Time Quantum's (Case-1)

5.1.6 Case-2: Non-Zero Arrival Times

In this case, CPU burst times are taken in random order and processes arrival times are assumed to be non-zero. A Ready queue with 5 processes P1, P2, P3, P4 and P5 has been considered as shown in table-3.

Table-3: Processes with their respective Arrival and Burst times (Case-2)

Process	Arrival Time	Burst Time
P1	0	7
P2	4	25
P3	10	5
P4	15	36
P5	17	18

5.1.7 Comparison Between Proposed Modified Round Robin & Traditional RR at Different Time Quantum's: (Case-2)

Table -4: Tabular Comparison Between Proposed Modified Round Robin & Traditional RR at Different Time Quantum's (Case-2)

Algorithm employed	Time Quantum	Avg. TAT	Avg. WT	Avg. RT
RR	5	47.80	29.60	4.00
Proposed RR	6	45.60	27.40	5.40
RR	7	43.80	25.60	6.80
RR	Calculated (STQ=7)	38.20	20.00	6.80
RR	8	45.60	27.40	6.40
RR	9	43.80	25.60	7.40

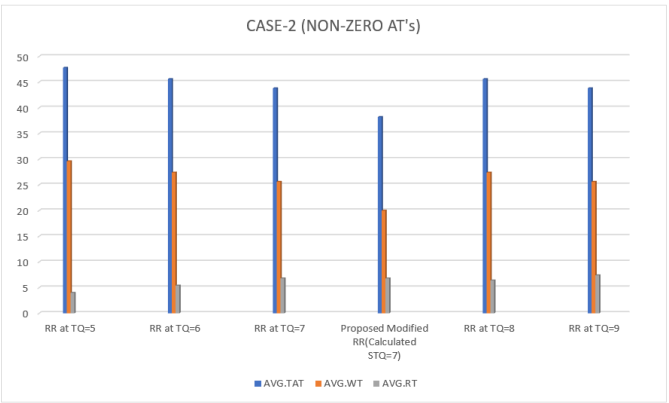

Chart-2: Graphical Comparison Between Proposed Modified Round Robin & Traditional RR at Different Time Quantum's (Case-2)

5.2 Comparison of the Proposed algorithm with IRR, AAIRR, ENHANCED RR, RRRT and ENRICHED RR Algorithm's

5.2.1 Assumptions

It has been assumed that the system where all these experiments are performed is Single processor system and all the processes have equal priority. It is also assumed that the Number of processes, Burst times and Arrival times are well-known even before submitting the processes to the processor. It is also assumed that the processes are only CPU bound.

5.2.2 Experiments Performed

Three different cases have been taken for performance evaluation of our proposed algorithm.

- In Case-1, CPU burst times are taken in random order and processes arrival times are assumed to be zero.
- In Case-2, CPU burst times are taken in increasing order and processes arrival times are assumed to be zero.
- In Case-3, CPU burst times are taken in decreasing order and processes arrival times are assumed to be zero.

5.2.3 Dataset Description

Test cases are adopted from the paper published by Afaf Abd Elkader and Kamal Eldahshan [7].

5.2.4 Case-1: Random Burst Times

In this case, CPU burst times are taken in Random order and processes arrival times are assumed to be zero. A Ready queue with 5 processes P1, P2, P3, P4 and P5 has been considered as shown in table-5.

Table-5: Processes with their respective Arrival and Burst times (Case-1)

Process	Arrival Time	Burst Time
P1	0	24
P2	0	40
P3	0	5
P4	0	12
P5	0	34

5.2.5 Comparison Between Proposed Modified Round Robin and Various Modified RR's (Case-1)

Table-6: Tabular Comparison Between Proposed Modified Round Robin and Various Modified RR's (Case-1)

Algorithm employed	Time Quantum	Avg. TAT	Avg. WT	Avg. RT
Proposed Modified RR	Calculated (STQ=7)	60.40	37.40	15.40
IRR	7	67.40	44.40	14.20
AAIRR	7	63.20	40.20	14.80
ENHANCED RR	6	66.20	43.20	12.80
RRRT	Calculated (TQ=11)	59.40	36.40	17.80
ENRICHED RR	Calculated (TQ=17)	54.00	31.00	24.20

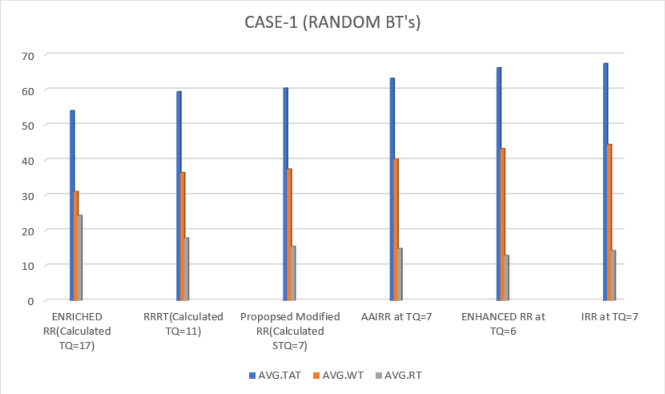

Chart-3: Graphical Comparison Between Proposed Modified Round Robin & Various Modified RR's (Case-1)

5.2.6 Case-2: Increasing Burst Times

In this case, CPU burst times are taken in Increasing order and processes arrival times are assumed to be zero. A Ready queue with 5 processes P1, P2, P3, P4 and P5 has been considered as shown in table-7.

Table-7: Processes with their respective Arrival and Burst times (Case-2)

Process	Arrival Time	Burst Time
P1	0	7
P2	0	10
P3	0	20
P4	0	26
P5	0	30

5.2.7 Comparison Between Proposed Modified Round Robin and Various Modified RR's (Case-2)

Table-8: Tabular Comparison Between Proposed Modified Round Robin and Various Modified RRs (Case-2)

Algorithm employed	Time Quantum	Avg. TAT	Avg. WT	Avg. RT
Proposed Modified RR	Calculated (STQ=3)	59.60	41.00	06.00
IRR	3	59.60	41.00	06.00
AAIRR	3	59.60	41.00	06.00
ENHANCED RR	2	61.00	42.40	04.00
RRRT	Calculated (TQ=9)	48.80	30.20	17.00
ENRICHED RR	Calculated (TQ=13)	46.00	27.40	22.20

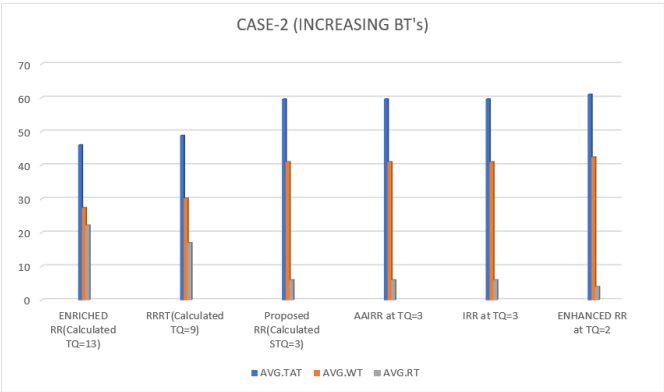

Chart-4: Graphical Comparison Between Proposed Modified Round Robin & Various Modified RR's (Case-2)

5.2.8 Case-3: Decreasing Burst Times

In this case, CPU burst times are taken in Decreasing order and processes arrival times are assumed to be zero. A Ready queue with 5 processes P1, P2, P3, P4 and P5 has been considered as shown in table-9.

Table-9: Processes with their respective Arrival and Burst times (Case-3)

Process	Arrival Time	Burst Time
P1	0	45
P2	0	36
P3	0	30
P4	0	18
P5	0	10

5.2.9 Comparison Between Proposed Modified Round Robin and Various Modified RR's (Case-3)

Table-10: Tabular Comparison Between Proposed Modified Round Robin and Various Modified RR's (Case-3)

Algorithm employed	Time Quantum	Avg. TAT	Avg. WT	Avg. RT
Proposed Modified RR	Calculated (STQ=5)	85.80	58.00	14.00
IRR	5	101.80	74.00	10.00
AAIRR	5	89.80	62.00	13.00
ENHANCED RR	4	99.40	71.60	8.00
RRRT	Calculated (TQ=13)	73.60	45.80	26.60
ENRICHED RR	Calculated (TQ=20)	65.80	38.00	38.00

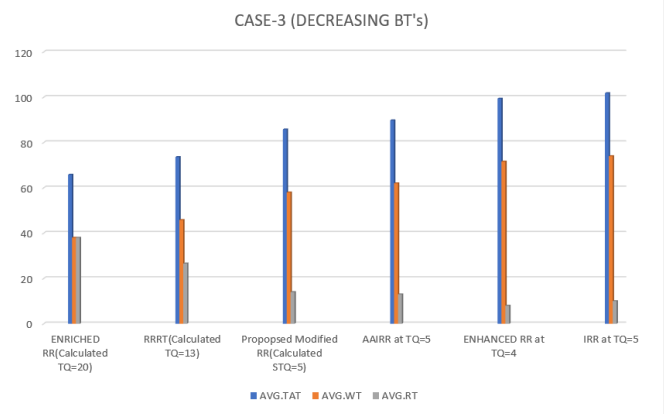

Chart-5: Graphical Comparison Between Proposed Modified Round Robin & Various Modified RR's (Case-3)

6. RESULT AND ANALYSIS

One of the prime tasks of the operating system is the allocation of CPU to the processes which are waiting for execution. Many CPU scheduling algorithms have been introduced, which have both advantages and disadvantages. The proposed modification in Round robin scheduling algorithm with calculation of smart time quantum is giving better performance than conventional Round Robin algorithm which is shown by taking 2 different cases to ensure a fair comparison between the Proposed approach and Traditional Round Robin approach, we even adopted the Test cases for these cases from a valid source of journal [6]. In both the cases, average waiting time, average turnaround time and average response time have been reduced greatly in the proposed approach and hence the system performance has been improved. Even the proposed modification in the Round robin algorithm with calculation of smart time quantum is giving better performance than existing modified versions of Round Robin algorithm which is shown by taking 3 different cases to ensure a fair comparison between the proposed approach and existing modified versions of Round Robin algorithm, we even adopted the Test cases for these cases from a valid source of journal [7].

In Case-1, where burst times of all the processes are taken in random order, we find that the proposed approach is executing better than IRR, AAIRR, and ENHANCED RR in terms of Average Waiting Time, Average Turn Around Time, and Average Response Time; although both RRRT and ENRICHED RR are better than the proposed approach in terms of both Average Waiting Time and Average Turn Around Time, but the proposed approach is far better than the both RRRT and ENRICHED RR in terms of Average Response Time. Here, even Average Response Time is also an important factor of measuring efficiency because conventional Round Robin finds its specialty and uniqueness among other conventional scheduling algorithms mainly because of its minimum Average Response Time.

In Case-2, where burst times of all the processes are taken in increasing order, we find that the proposed approach is executing equally efficient to IRR and AAIRR, and executing better than ENHANCED RR in terms of Average Waiting Time, Average Turn Around Time, and Average Response Time; although both RRRT and ENRICHED RR are better than the proposed approach in terms of both Average Waiting Time and Average Turn Around Time, but the proposed approach is far better than both RRRT and ENRICHED RR in terms of Average Response Time.

In Case-3, where burst times of all the processes are taken in decreasing order, we find that the proposed approach is executing better than IRR, AAIRR, and ENHANCED RR in terms of Average Waiting Time, Average Turn Around Time, and Average Response Time; although both RRRT and ENRICHED RR are better than the proposed approach in terms of both Average Waiting Time and Average Turn Around Time, the proposed approach is far better than both RRRT and ENRICHED RR in terms of Average Response Time.

7. FUTURE SCOPE

Cloud computing is one of the prime technologies we are using in recent times, it allows users (individuals or organizations) to access computational resources like: Software and Hardware as services remotely through the Internet. As cloud computing is serving millions of users parallelly, it must have the ability to meet all user requests with high efficiency and guarantee of Quality of service (QOS). Thus, we need to implement an appropriate CPU scheduling algorithm to efficiently and fairly meet all these requests of users simultaneously. As the most well-known Conventional Round Robin algorithm is executing less efficiently when compared to the proposed approach, the proposed approach can be implemented in this field with making little modifications related to Cloud computing domain.

8. CONCLUSIONS

Many algorithms are proposed to improve the efficiency of Traditional Round Robin algorithm to minimize the average waiting time, average turnaround time and average response time. Time Quantum is the most important parameter in the performance of these proposed algorithms.

We can conclude from the above experiments that the proposed algorithm performs better than the previously developed approaches in terms of performance parameters such as average waiting time, average turnaround time and average response time.

But we can't generalize and say that a particular algorithm is best, even a best algorithm gives worst results in some drastic cases. But as per our experimental analysis the proposed approach is performing very well when compared to Traditional round robin in almost all the cases we performed. Also, our approach is performing better than atleast four out of the five modified versions of Round Robin algorithm: IRR, AAIRR, ENHANCED RR, RRRT and ENRICHED RR in each and every different case we have performed.

REFERENCES

[1] Neha and A. Jiyani, "An Improved Round Robin CPU Scheduling Algorithm," IRE Journals, vol. 1, no. 9, pp. 82-86, 2018.

[2] S. Aliyu, S. E. Abdullahi, A. M. Mustapha and S. E. Abdullahi, "An Additional Improvement in Round Robin (AAAIRR) CPU Scheduling Algoritm," International Journal of Advanced Research in Computer Science and Software Engineering , vol. 4, no. 2, pp. 601-610, 2014.

[3] J. Khatri, "An Enhanced Round Robin CPU Scheduling Algorithm," IOSR Journal of Computer Engineering (IOSR-JCE) , vol. 18, no. 4, pp. 20-24, 2016.

[4] A. Sharma and G. Kakhani, "Analysis of Adaptive Round Robin Algorithm and Proposed Round Robin Remaining Time Algorithm," International Journal of Computer Science and Mobile Computing, vol. 4, no. 12, pp. 139-147, 2015.

[5] K. ElDahshan, A. A. Elkader and N. Ghazy, "Achieving Stability in the Round Robin Algorithm," International Journal of Computer Applications, vol. 172, pp. 15-20, 2017.

[6] D. F. Rashid and M. Kumar, "An Improved Round Robin CPU Scheduling Algorithm with Varying Time Quantum," International Journal of Computer Science, Engineering and Applications (IJCSEA) , vol. 4, no. 4, pp. 1-8, 2014.

[7] K. Eldahshan and A. A. Elkader, "Round Robin based Scheduling Algorithms, A Comparative Study," Automatic Control and System Engineering Journal, vol. 17, no. 2, pp. 29-42, 2017.

BIOGRAPHIES

Mr. Edula Vinay Kumar Reddy is pursuing B.Tech (CSE) from Vellore institute of Technology, Vellore and presently is in 3rd Year. His research interests are Operating Systems, Data Science and Artificial Intelligence.

Mr. Kotha V V S Aakash is pursuing B.Tech (CSE) from Vellore institute of Technology, Vellore and presently is in 3rd Year. His research interests are Software Development, Machine Learning and Operating Systems.