Improvised Dynamic Round-Robin Scheduling for Optimum Resource Utilization in Cloud Systems

Suvarna N.A.
Research Scholar, GD Goenka University
Gurugram, Haryana -122102, India

Rashmi Priya
Assistant Professor, GD Goenka University
Gurugram, Haryana -122102, India

ABSTRACT
Efficient task scheduling, aimed at maximizing resource utilization (such as CPU, memory, and bandwidth) while executing a maximum number of tasks, is crucial in large-scale cloud computing and associated architectures (e.g., Fog/Edge/IoT). These architectures are employed to support new business models and ensure uninterrupted services, even with intermittent connections to cloud servers. Resource optimization plays a vital role in determining the quality of service (QoS) provided to customers. Furthermore, task scheduling for parallel processing is fundamental for comprehending resource utilization, inter-process communication, network latency, load balancing, job migration, and fault tolerance. This research paper endeavours to explore, analyse, design, and implement scheduling algorithms that optimize the utilization of computing resources. The newly developed algorithm exhibits improved performance compared to existing ones. The results are interpreted and substantiated based on various QoS indicators.

General terms
Algorithms

Keywords
Round-Robin (RR); Burst Time (BT); Time Slice (TS); Shortest Job First (SJF), Average Completion Time (ACT) ; Average Waiting Time (AWT).

1. INTRODUCTION
In Cloud Computing, efficient allocation of CPU time to incoming tasks is a crucial resource optimization challenge. The Datacentre Broker is responsible for dispatching incoming tasks to Virtual Machines, and each Virtual Machine executes the assigned tasks according to a scheduling strategy. The scheduling algorithm plays a vital role in resource management, and Round-Robin Scheduling is a widely popular and traditional scheduling algorithm.

Round-Robin Scheduling is a pre-emptive algorithm that divides the CPU time into small, fixed intervals known as Time-Slices (TS) or Quantum-Time (QT). In each iteration, every process in the ready queue is allocated a single TS. If the process is completed within a TS, it gets terminated; otherwise, it gets added to the end of the queue with its remaining Burst Time (BT). The CPU iterates through the processes in multiple cycles until all tasks are completed.

The effectiveness of Round-Robin Scheduling is governed by two major concerns:

a) The value of Time-Slice relative to Burst-Time.
b) Adjustment of Time-Slice to suit the unpredictable dynamic loads in Cloud Systems.

Keeping the Time-Slice small causes additional overhead due to Context Switching (CS), which involves storing the context of each task in the stack and wastes precious CPU time. Conversely, setting the Time-Slice too high increases the average completion time (ACT) and worsens the performance by making shorter jobs wait longer behind longer ones. Therefore, an optimal Time-Slice is crucial for designing an efficient Round-Robin algorithm.
Furthermore, the dynamic nature of cloud computing demands a dynamic scheduling policy that can adapt to the ever-changing load.

2. RELATED WORKS
Numerous scheduling algorithms have been designed and developed by researchers to achieve optimal CPU utilization and reduce process waiting time. This paper focuses on improving the Round Robin (RR) Scheduling algorithm. Different formulae have been proposed by previous researchers to determine the value of time slice (TS) and to incorporate the dynamic nature of computing platforms such as Fog/IoT/Edge. This is achieved by varying the TS after each round of execution. Performance evaluation and results assessment are carried out by measuring several parameters, including Average Completion Time (ACT), Average Waiting Time (AWT), MakeSpan, Context Switches (CS), and Average Response Time (ART).

Pandaba Pradhan et al. [1] proposed an improved dynamic RR algorithm. The first task cycle has TS equal to the first task, and for all subsequent iterations, the TS is equal to the average of the burst-times of the remaining jobs in the queue. A comparison was made to show the improvement over the static RR algorithm.

Saqib Ul Sabha et al. [2] suggest sorting tasks based on their burst times and using a random TS at the initial stage. The algorithm compares the TS with the remaining burst time of each task. If the remaining burst time after execution is less than or equal to half the TS, the process is executed completely. Sorting is done after each iteration, but only when a new process enters the queue.

Linz Tom. and Bindu V.R. [3] sort the tasks and virtual machines (VMs) in decreasing order of computing time and apply an improved algorithm, Dynamic Task Scheduling Based on Completion Time (DTBCT). The authors recommend maintaining multiple queues and attaching tasks to VMs in sorted order. Tasks are migrated based on completion time in dynamic mode. Time-Slice is dynamically determined based on the burst time of the task under consideration. An improvement in all performance metrics, including AWT, ACT, CS, and Make Span, is achieved in comparison with static RR, First Come First Serve (FCFS), and Shortest Job First (SJF) algorithms.

Sakshi et al. [4] elaborates various methods of the RR scheduling with dynamic TS, including the Average Median RR (AMRR), A New Round Robin (ANRR), and Modified Median RR Algorithm (MMRRA). The authors then propose a new enhanced algorithm for calculating dynamic TS as Median Average Round Robin (MARR). The authors practically prove that MARR performs better than earlier schemes.

- **TS = (Av. BT + Highest BT)/2 (AMRR) - (1)**
- **TS = Average BT (ANRR) - (2)**
- **TS = √ median * Highest BT (MMRRA) - (3)**

The author then suggests the new improved algorithm by calculation of dynamic time slice as:
- **TS = (Av. BT + Median BT) /2 (MARR) - (4)**

Abdulaziz et al. [5] proposed four different algorithms based on the calculation of TS, including Optimum Round-Robin using Manhattan Distance (ORRMD), Improved RR Algorithm (IRR), Adaptive RR Algorithm (ARRA), and Best Time Quantum RR (BTQRR). Although the author claims better results for ORRMD and ARRA, the results reveal that the number of context switches is significantly higher in ORRMD, and the average response time is longer in ARRA when compared with other algorithms.

- **TS = (Max BT + Min BT) (ORRMD) - (5)**
- **TS =√ (Median * Highest)BT (IRR) - (6)**
- **TS =Median BT (ARRA) - (7)**
- **TS =(Mean + Median)BT/2 (BTQRR) - (8)**

Uferah Shafi et.al [6] conducted a study on various enhancements to the traditional Round Robin (RR) algorithm and compared them with their devised algorithm. The paper mentions the following variations of RR:
Improved Round Robin (IRR): The first cycle follows RR, and subsequent cycles use Shortest Job First (SJF).
Optimum Multilevel Dynamic Round Robin (OMDRR): This method involves the following steps:
- Ordering of processes
- Application of an intelligent time slice
- Doubling of time slice after each iteration
- Application of a condition to reduce waiting time based on remaining burst time

Priority based Round Robin (PRR): In the first iteration, processes are executed according to priority. Subsequent iterations arrange processes based on remaining burst time for execution.
Uferah Shafi et.al [6] then proposes the following approach:
1) Set the time slice (TS) to the lowest BT.
2) If (TS<Threshold), set TS to Threshold.
3) Perform the following steps repeatedly:
  3.1) Execute the first task in the RQ.
  3.2) If (Remaining BT < TS/2), pre-empt the process.
  3.3) Else, place the process at the end of RQ.

Sanaj M S, Dr. Joe Prathap P M [7], in their literature study mentions the nature inspired "Ant Colony Optimization" (ACO), "Genetic Algorithm" (GA), "Multiple Pheromone Algorithm" (MPA) which is a variation of ACO, FCFS, Particle Swarm Optimization (PSO). The author suggests the simple setting of time slice as mean value, which in normal distribution of data tends to be the median itself.
In [8], Shihab Ullah et al. refer to an algorithm called **ODTSRR (Optimum Dynamic Time Slicing Round Robin Scheduling)**, where TS is chosen as the median of the burst times in the task queue. After executing each task, if the remaining BT is less than TS, the task is fully executed to reduce waiting time. The authors then propose a different algorithm called **IODTSRR (Improved Optimum Dynamic Time Slicing Round Robin Scheduling Algorithm)**, where the job is executed to completion if the BT is less than or equal to twice the TS.

In [9], Rahul Mishra et al. propose the **"Improved Round Robin Algorithm for effective Scheduling Process for CPU"** which sets TS to the average BT and executes the job to completion if the remaining BT is less than TS. The literature study primarily focuses on selecting TS as mean, median, (mean + median)/2, (mean + maximum)/2, √(Median * Highest BT), and combining Round Robin with SJF to reduce waiting time. Most of the referenced research works with small data sets, as outlined in Table .

Komal Mahajan et al. [13] propose saving the earlier allocation state of a virtual machine (VM) to a request from a specific user, providing server affinity to the tasks. Table-1 summarizes the literature study.

3. PROBLEM STATEMENT

Load balancing in distributed systems is crucial for enhancing execution speed and optimizing resource utilization. Load balancing is required during various stages, such as when allocating a data center, allocating a VM within a data center, and allocating CPU time for jobs within a VM [12][14]. This research focuses on load balancing within a VM through scheduling algorithms, considering the third stage mentioned above. Inspired by previous research efforts to improve the Round Robin algorithm for modern computing systems with Edge, Fog, and IoT architecture, this study explores dynamic algorithms for further advancements and improved performance.

Table 1: Comparison of literature studied on Improvisation of Round Robin Scheduling

Ref.	Algorithm	Dynamic /Static	Time-Slice/Algorithm	Sorting of Jobs	No. of Tasks	Limitations	Comparisons
[1]	Modified Round-Robin	Dynamic	TS = Average BT	No	5	Lacks the benefits of SJF as sorting is not done.	RR

[2]	Improved Round-Robin with Intelligent Time Quantum based on remaining BT	Dynamic	TS = Random value
			If (BT <= 1.5*TS)
			Execute the complete process
			Else
			Execute for Time = TS
			End If
				Yes	4	No improvement in turn around time when processes arrive at different times.	RR

[3]	Dynamic Task Scheduling Based on Completion Time(DTBCT)	Dynamic	TS = Average Burst Time	Yes	10 to 1000	Main work revolves around managing multiple queues based on burst time.	RR SJF FCFS
[4]	Median Average Round-Robin Algorithm (MARR)	Dynamic	TS = (Average + Median)/2	Yes	4, 6	Turn Around Time is not improved when compared to ANRR and no change in context switching.	ANRR AMRR MMRRA
[5]	1. Optimum Round-Robin using Manhattan Distance 2. Improved Round-Robin Algorithm 3. Adaptive Round-Robin Algorithm Best Time Quantum Round-Robin CPU Scheduling Algorithm	Dynamic Dynamic Dynamic Dynamic	TS = (Max BT + Min BT) (ORRMD) TS = √ Median * Highest BT (IRR) TS = Median (ARRA) TS = (Mean + Median)/2 (BTQRR)	No Yes Yes Yes	10000	Average Response Time, Average Waiting Time and Average turn Around Time is better in ORRMD. Context Switching is better in ARRA	ORRMD ARRA

[6]	Amended Dynamic Round Robin	Dynamic	TS = Lowest BT
			If (TS < Threshold)
			TS = Threshold
			End If
			Do
			{ Execute first task in RQ
			If (Remaining BT < TS/2)
			Pre-empt the process
			Else
			Place the process in the end of RQ
			End If
			} While (RQ != NULL)
				Yes	5	The results in terms of Average Waiting Time, Average Completion Time, Context Swithing got improved in the order of RR->PRR->OMDRR->IRR->ADRR	RR PRR OMDRR IRR ADRR

[7]	An Enhanced Round Robin (ERR) algorithm	Dynamic	TS = Mean BT	Yes	7	Compares with PSO, GA, ACO,	PSO GA ACO

	for Effective and Efficient Task Scheduling in cloud environment					MPA, FCFS, Min-Min. But comparison is only on makespan and energy levels. AWT is compared with only RR	MPA FCFS Min-Min

[8]	Improved Optimum Dynamic Time Slicing Round Robin Scheduling Algorithm (IODTSRR)	Dynamic	TS = Median BT
			If (Remaining BT < TS)
			Execute the complete Process
			End If
				Yes	5, 8	Context Switching has increased in IODTSRR when processes (7) arrive at different times.	ODTSRR

[9]	Improved Round Robin Algorithm for effective Scheduling Process for CPU	Dynamic	TS = Average BT
			If (Remaining BT < TS)
			Execute the complete process
			End If
				Yes	5	Improved over RR and CRR.	RR CRR

4. PROPOSED WORK

Authors propose an improved version of the Round-Robin algorithm that surpasses previous research studies. Our proposed algorithm combines the strengths of Shortest Job First (SJF) and Round-Robin (RR) approaches, making it highly suitable for modern large-scale and unpredictable computing requirements.

To achieve optimal performance, the paper introduces variations in the time slice of the Round-Robin algorithm. Through extensive experimentation, it is discovered that the best outcomes were achieved by integrating SJF principles, which involve ordering tasks in increasing order of their burst times. Additionally, we determined that setting the time slice to a value that would clear the queue within a single iteration yielded favourable results. Thus, we propose the Optimal Time Slice(OTS) as:

OTS = BT of the second-largest job in the queue - (9)

Based on this Optimum Time Slice (OTS) concept, we present a novel algorithm called **"Novel Optimum Dynamic Approach to Round Robin Scheduling" (NODARR)**. The algorithm's steps are as follows:

- Sort the processes in ascending order of BT.
- Insert the sorted processes into the RQ
- Set the time slice to the second-largest BT (OTS) in the RQ.
- Initialize the completion time and waiting time and context switches of each process to 0.
- Set the iteration counter to 1 & enter a loop.
- While the RQ is not empty, perform the following steps:
- For each task in the RQ, do the following:

If the BT of the task is ≥TS:
Execute the task for one TS.
Update the completion time by adding TS.
Update the waiting time by adding TS.
Increment the context switches by 1.
Decrease the remaining BT by TS.
Insert the process at the end of the RQ.
Set BT to the remaining burst time.
Else (if BT ≤TS):
Execute the task for its entire BT.
Update the completion time by adding the BT.
Update the waiting time by adding the BT.
Increment the context switches by 1.
Set the remaining BT to 0.
Remove the process from the RQ.

- End the if-else condition.
- End the loop for each task.
- Rearrange the tasks in the RQ in the order of their BT to accommodate new entries.
- Update the time slice to the OTS.
- Increment the iteration counter by 1.
- Repeat the do-while loop.
- End the do-while loop when the termination condition is met.

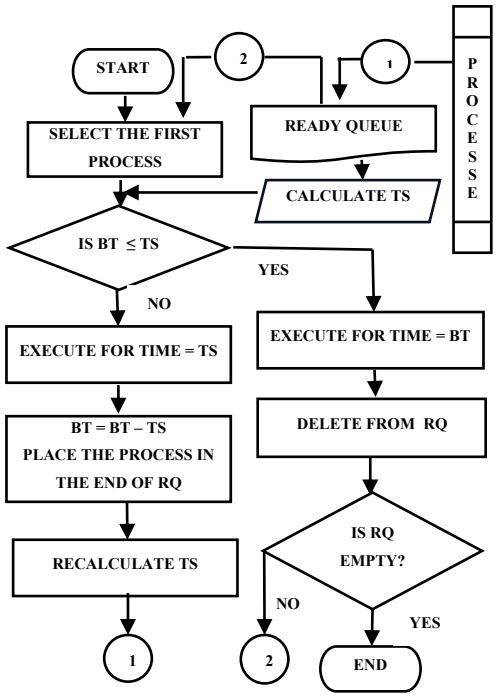

Figure 1: Flow Chart of the Proposed Algorithm-NODARR

The proposed algorithm, NODARR, demonstrates the capability to optimize round-robin scheduling by dynamically adjusting the time slice based on the second-largest burst time in the queue. By combining SJF principles and effectively managing the execution of tasks, the algorithm significantly reduces waiting time.

5. EXPERIMENTAL SETUP

The experimental setup in CloudSim consists of the following components:

a) Datacentre b) Datacentre broker
c) Hosts d) Processing Elements
e) Virtual Machines f) Cloudlets

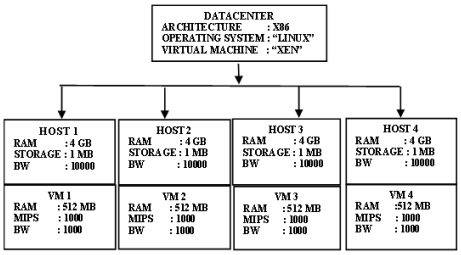

Figure 2: Experimental Setup

The number of Processing Elements (CPUs) used is 1. The Datacentre specifications included an "x86" architecture, Linux operating system, and Xen virtual machine. The Cloudlet specifications involved 40 cloudlets with random Burst Times ranging from 1 to 100. There were 4 Hosts, each with one processing element, having 4096 MB RAM, 1000000 storage, and 10000 bandwidths. Additionally, 4 Virtual Machines were created with 512 MB RAM, 1000 MIPS, and 1000 bandwidth.

The simulator used for the experiment is CloudSim, and the dataset employed is the Google Dataset available at the following link:
GoogleDataset (https://zenodo.org/record/3696775#.YxtXFHZBy3A). This dataset focused on "Problem instances for scheduling jobs with time windows on unrelated parallel machines."

The experiment was conducted in two phases, where different cases were explored. The cases were as follows:

Case 1: Static RR with TS of 5 seconds.
Case 2: Static RR with TS of 10 seconds.
Case 3: Static RR with TS of 20 seconds.
Case 4: Static RR with TS = AMRR (1).
Case 5: Static First-Come, First-Served (FCFS) .
Case 6: Static SJF + RR with TS = AMRR (1).
Case 7: Dynamic RR with TS = AMRR (1).
Case 8: Dynamic SJF + RR with TS = AMRR (1)

The burst times for the 40 tasks allocated to the 4 virtual machines in groups of 10 each are presented in Table 2 below.

Table 2: Burst Times of 40 Processes in Millions of Instructions (Mis)

	PROCESS LENGTHS IN MIs			
PROCESS	**VM-1**	**VM-2**	**VM-3**	**VM-4**
P1	32	20	54	49
P2	83	68	110	45
P3	26	92	34	76
P4	30	18	24	40
P5	86	61	55	51
P6	41	101	21	72
P7	99	33	55	50
P8	69	84	48	94
P9	28	36	77	47
P10	25	59	55	104
Average	**51.90**	**57.20**	**53.30**	**62.80**

6. RESULTS OF PHASE-I

When subjecting the 40 tasks from Table 2 to the eight different cases across 4 virtual machines (VMs), the results for Average Waiting Time (AWT), Active Time (ACT), and Completion Status (CS) were recorded and presented in Table 3.

Table 3: Results of Phase-1

CASE	VM 1			VM 2		
	ACT	AWT	CS	ACT	AWT	CS
1	369	317	108	408	351	119
2	366	314	56	402	345	62
3	376	324	32	384	327	33
4	346	294	13	359	302	13
5	289	237	10	306	249	10
6	234	182	13	258	201	13
7	346	294	13	359	302	13
8	234	182	14	258	201	14
CASE	**VM 3**			**VM 4**		
	ACT	**AWT**	**CS**	**ACT**	**AWT**	**CS**
1	397	344	109	505	442	238
2	403	349	58	496	433	67
3	393	340	31	485	422	36
4	312	259	11	325	262	12
5	330	277	11	313	250	10
6	228	175	11	297	234	12
7	312	259	11	325	262	12
8	228	175	11	297	234	13

6.1 Result Analysis of Sample Cases

6.1.1 *Case-3 with TS=20 under VM-3*

A specific sample, namely Case-3 under VM-3, was selected for further analysis. The Gantt Chart in Figure 3 illustrates the execution timeline of the 10 processes running under VM-3 for Case-3, which employed Round Robin (RR) scheduling with a time slice (TS) of 20 seconds.

The completion times of 10 processes recorded while running on VM-3 for Case-3, utilizing a time slice (TS) of 20 seconds, are presented in Table 4.

Average Completion Time:

(373+533+254+258+408+279+423+431+503+466)/10 = 3928/10 = 392.8

Average Waiting Time (AWT): Average Waiting Time is defined as:

AWT = Sum of ((Time of Completion - Burst Time) of each job) / Number of Jobs

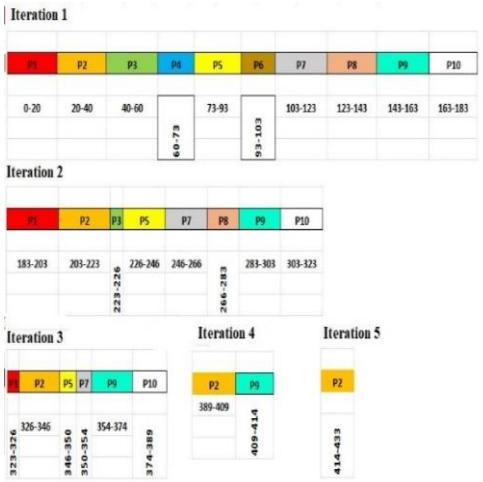

Figure 3: Gantt Chart of Case-3 with Time Slice = 20 under VM-3

Table 4: Completion Times of 10 processes under VM-3 (Case-3)

Process	Completion Times
P1	373
P2	533
P3	254
P4	258
P5	408
P6	279
P7	423
P8	431
P9	503
P10	466

So, for the above example,
AWT = (319 + 423 + 220 + 234 + 353 + 258 + 368 + 383 + 426 + 411) / 10 = 3395/10 = 339.5

Context Switches: Number of Context Switches is defined as

Context Switches (CS) = Sum of (Number of Iterations of Each Process)

CS = 3 + 6 + 2 + 2 + 3 + 2 + 3 + 3 + 4 + 3 = 31

6.1.2 Case-8 with TS= AMRR under VM-1

Table 5 below lists the tasks and related averages and time-slice values in each iteration.

Table 5: Sorted Burst Times under VM 1

Burst Times	Dynamic Averages	Time Slices
25, 26, 28, 30, 32, 41, 69, 83, 86, 99	51.9, 14,5	75.45,19, 5

Iteration 1:
Average = 25 + 26 + 28 + 30 + 32 + 41 + 69 + 83 + 86 + 99
= 51.9
TS = (Average BT + Highest BT)/2
= (51.9 + 99)/2 = 75.45

As TS is 75.45, seven tasks are executed completely and remaining 3 tasks are partly executed. The remaining burst times of these three tasks are 8, 11 and 24.

Iteration 2:
Av = (8 +11+24)/3 = 14
TS = (14 +24)/2 = 19
Now only last task is left incomplete with remaining burst time as 5.

Iteration 3:

Av = 5, TS = (5+5)/2 = 5

Average Completion Time :

(25 + 51 + 79 + 109+ 141 + 182 + 251 + 484 + 495 + 519) / 10 = 233.6

Average Waiting Time:

(0 + 25 + 51 + 79 + 109 + 141 + 182 + 401 + 409 + 420)/10 = 181.7

Total number of Context Switches:

(1 + 1 + 1 + 1 + 1 + 1 + 1 + 2 + 2 + 3) = 14

The total number of context switches in each VM under the above mentioned eight different cases of different time-slices is given in the graph of figures 4 and 5. From the graphs, it is evident that the context switching is minimum for case-5 (FCFS).

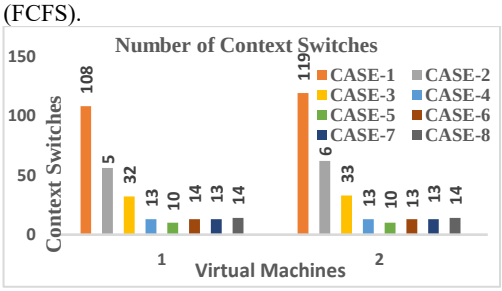

Figure 4:Context Switches under VM1 &VM2

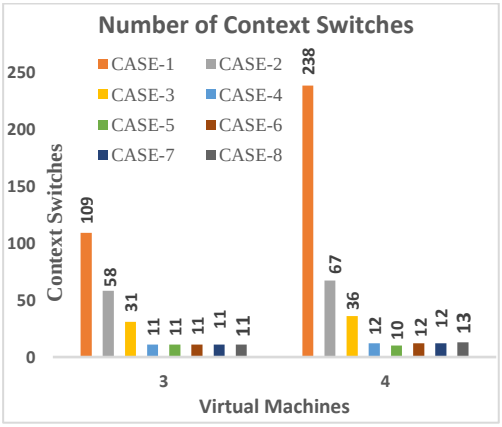

Figure 5:Context Switches under VM-3, VM-4

But, considering the graphs for completion times as shown in figures 6 and 7 and the graphs for waiting times as shown in figures 8 to 11, best results are obtained for the case-7 and case-8. Therefore, these cases are short-listed for phase-II, where they are compared with the proposed algorithm.

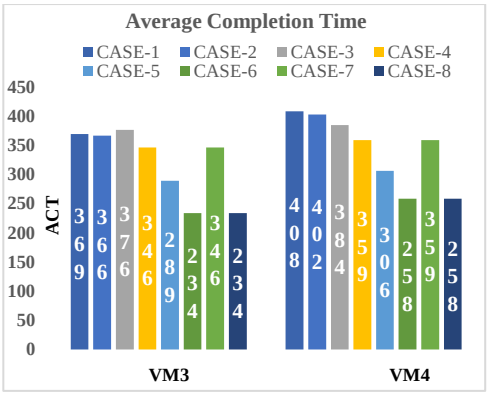

Figure 6: ACT under VM1 and VM2

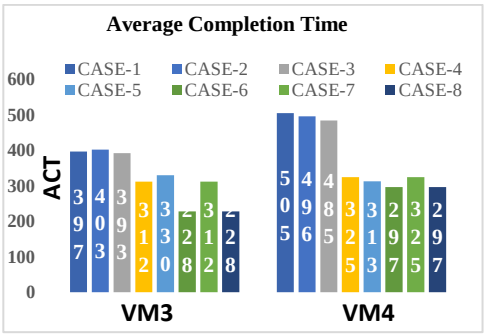

Figure 7: ACT under VM3 and VM4

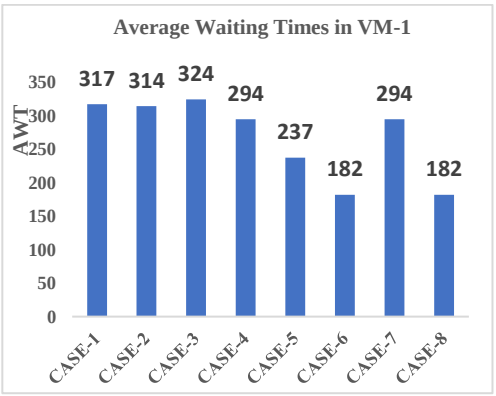

Figure 8: Average Waiting Times under VM1

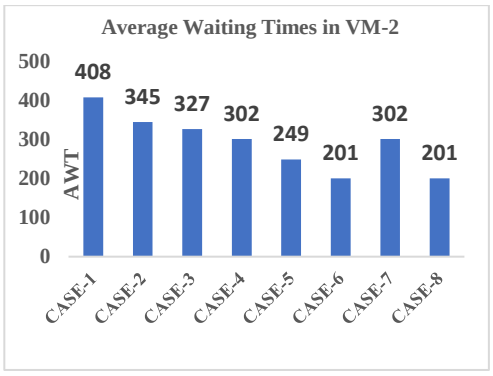

Figure 9:Average Waiting Times under VM2

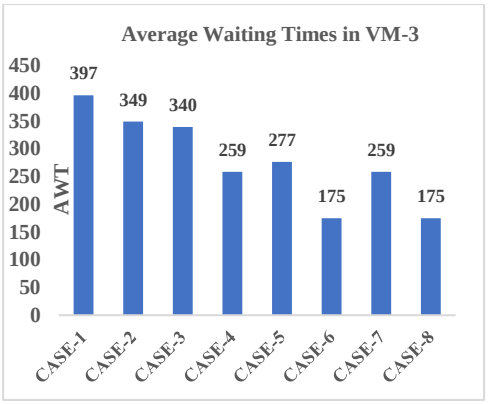

Figure 10:Average Waiting Times under VM3

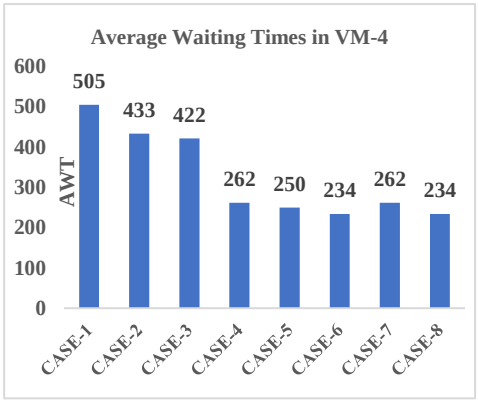

Figure 11:Average Waiting Times under VM4

7. RESULTS OF PROPOSED ALGORITHM

In phase-II of the experiment, the selected cases from phase-I that yielded the best results, namely Round-Robin with Shortest Job First (case 6 and case 8), were further examined. The time slice was varied based on recommendations from previous researchers. Subsequently, these cases were compared against the proposed algorithm to demonstrate the superior performance of the proposed approach.

7.1 Performance Analysis

Table 6 below presents a collection of 40 processes that were executed together under a single virtual machine (VM) utilizing the proposed algorithm. The outcomes achieved from this execution were then compared with the results obtained using other algorithms referenced in [4].

Table 6: 40 Processes and their burst times

Process	BT	Process	BT
P1	21	P21	43
P2	72	P22	99
P3	15	P23	23
P4	19	P24	13
P5	75	P25	44
P6	30	P26	10
P7	88	P27	44
P8	58	P28	37
P9	17	P29	65
P10	11	P30	55
P11	9	P31	38

P12	57	P32	34
P13	81	P33	65
P14	7	P34	29
P15	50	P35	40
P16	90	P36	61
P17	22	P37	39
P18	73	P38	83
P19	25	P39	31
P20	48	P40	70

The obtained results for the proposed NODARR algorithm, along with the algorithms referenced in [4], are organized and presented in Table 7. Corresponding graphical representations can be found in Figures 9 and 10.

Table 7: Results of the proposed algorithm

Time Slice	Algorithm	CS	ACT	AWT
(Av+median)/2	MARR (4)	73	851	806
Av+highest)/2	AMRR(1)	48	678	633
Second Largest	NODARR (9)	40	627	582

The results conclusively demonstrate that the proposed algorithm outperforms the other two algorithms mentioned in reference [4].

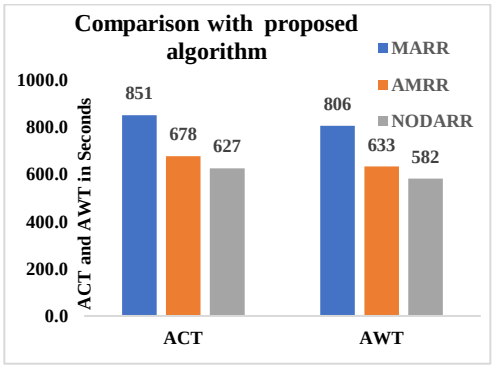

Figure 12: AWT and ACT of proposed algorithm (NODARR)

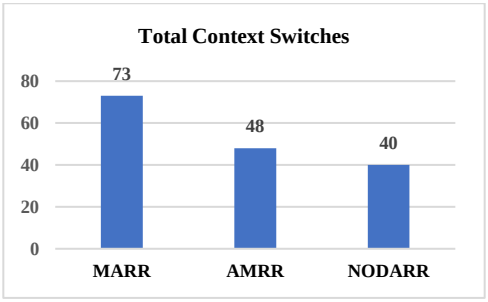

Figure 13: Context Switches of the proposed algorithm

7.2 Performance Analysis of the Proposed Algorithm with 1000 Processes

In order to further evaluate the improved performance of the proposed algorithm, the experiment was extended to involve the execution of 1000 processes. The outcomes of this extended experiment are illustrated in Figure 11. Once again, the proposed algorithm demonstrated superior performance compared to the other two algorithms across all three metrics: Average Waiting Time (AWT), Active Time (ACT), and Completion Status (CS).

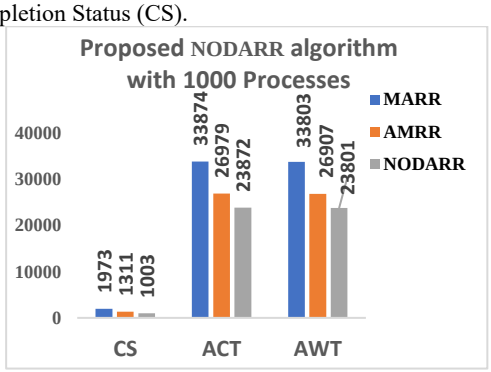

Figure 14: CS, ACT and AWT with large dataset (1000 processes)

7.3 Time Complexity of NODARR

Assuming that all tasks have equal lengths (l) and there are a total of n tasks, the proposed algorithm (NODARR) exhibits the following time complexity:

$$O(\sum_{i=0}^{n} l_i)$$

In contrast, the time complexity of the reference algorithm mentioned in [4] is given by:

$$O(\sum_{i=0}^{\log n}(\sum_{n-\frac{n}{2^i}}^{\frac{n}{2^{i+1}}} l_i))$$

8. CONCLUSION

The NODARR (Novel Optimum Dynamic Approach to Round Robin) algorithm, proposed in this study, outperformed previously suggested algorithms across different task lengths and data sizes. As a result, the study recommends the adoption of an optimal solution for dynamic time slice determination in round-robin scheduling:

Optimum Time Slice (OTS) = Burst Time of the second largest task

The proposed algorithm combines the advantages of both Round Robin (RR) and Shortest Job First (SJF). It leverages the dynamic time-slice feature of RR while incorporating the minimal waiting time aspect of SJF. In addition to these benefits, setting the time slice to the proposed optimal value minimizes the number of iterations and reduces context switching.

It should be noted that incoming task lengths can exhibit unpredictable variations, and the average value may not accurately represent the center of the data set. Therefore, using the average as a measure for calculating the time slice is not considered appropriate in such scenarios.

9. FUTURE SCOPE

To replicate the dynamic nature of the cloud environment, it is possible to simulate various configurations of virtual machines (VMs) and introduce different start times for jobs. Additionally, the same algorithm can be extended to facilitate inter-VM allocation of jobs, which aids in load balancing across multiple VMs. However, when considering inter-VM allocation, it becomes necessary to modify the algorithm to account for the latency involved in transmitting tasks between

VMs. A comprehensive algorithm can be devised at the load balancer level to effectively balance loads for both intra-VM and inter-VM allocation scenarios.

10. REFERENCES

[1] Pandaba Pradhan, Prafulla Ku. Behera, B N B Ray (2016), "Modified Round Robin Algorithm for Resource Allocation in Cloud omputing", Procedia Computer Science 85, 878 – 890

[2] Saqib Ul Sabha (2018), " A Novel and Efficient Round Robin Algorithm with Intelligent Time Slice and Shortest Remaining Time First", Materials Today Proceedings 5, 12009-12015.

[3] Linz Tom. and Bindu V.R. (2021), "Dynamic Task scheduling Based on Burst Time requirement for cloud environment", International Journal of Computer Networks & Communications (IJCNC) Vol.13, No.5, September.

[4] Sakshi, Chetan Sharma, Shamneesh Sharma, Sandeep Kautish, Shami A. M. Alsallami, E.M. Khalil, Ali Wagdy Mohamed (2022), "A new median-average round Robin scheduling algorithm: An optimal approach for reducing turnaround and waiting time", Alexandria Engineering Journal 61, 10527–10538.

[5] Abdulaziz A. Alsulami, Qasem Abu Al-Haija, Mohammed I. Thanoon, Qian Mao (2019), "Performance Evaluation of Dynamic Round Robin Algorithms for CPU Scheduling", DOI: https://doi.org/0.1109/SoutheastCon 42311.2019.9020439

[6] Uferah Shafi, Munam Shah, Abdul Wahid, Kamran Abbasi, Qaisar Javaid, Muhammad Asghar, and Muhammad Haider (2020), "A Novel Amended Dynamic Round Robin Scheduling Algorithm for Timeshared Systems", The International Arab Journal of Information Technology, Vol. 17, No. 1, January 2020.

[7] Sanaj M S, Dr. Joe Prathap P M, "An Enhanced Round Robin (ERR) algorithm for Effective and Efficient Task Scheduling in cloud Environment", https://doi.org/10.1109/ ACCTHPA49271.2020.9213198

[8] Shihab Ullah, "Improved Optimum Dynamic Time Slicing Round Robin Algorithm" (2017), 3rd International Conference on Electrical Information and Communication Technology (EICT), 7-9 December 2017, Khulna, Bangladesh.

[9] Rahul Mishra, Gaurav Mitawa, "Improved Round Robin Algorithm for effective Scheduling Process for CPU" (2021), Proceedings of the Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV 2021). IEEE Xplore Part Number: CFP21ONG-ART; 978-0-7381-1183-4.

[10] BING HU, (Senior Member, IEEE), FUJIE FAN, (Student Member, IEEE), KWAN L. YEUNG, (Senior Member, IEEE), AND SUGIH JAMIN (2018), "Highest Rank First: A New Class of Single-iteration Scheduling Algorithms for Input-queued Switches" IEEE Access, DOI: 10.1109/ACCESS.2017

[11] Mohammad Oqail Ahmad and Rafiqul Zaman Khan (2019), "Cloud Computing Modeling and Simulation using CloudSim Environment", International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8 Issue-2, July 2019.

[12] Shahbaz Afzal, G. Kavitha (2019), "Load balancing in cloud computing – A hierarchical taxonomical classification", Journal of Cloud Computing: Advances, Systems and Applications https://doi.org/10.1186/s13677-019-0146.

[13] Komal Mahajan, Ansuyia Makroo and Deepak Dahiya (2013) "Round Robin with Server Affinity: A VM Load Balancing Algorithm for Cloud Based Infrastructure Journal of Information Processing Systems", DOI: 10.3745/JIPS.2013.9.3.379.

[14] Altaf Hussain, Muhammad Aleem , Muhammad Arshad Islam, Muhammad Azhar Iqbal (2018), "A Rigorous Evaluation of State-of-the-Art Scheduling Algorithms for Cloud Computing", IEEE Access, DOI: 10.1109/ACCESS.2018.2884480.

[15] Mung Chiang, *Fellow, IEEE*, and Tao Zhang, *Fellow, IEEE* (2016), "Fog and IoT: An Overview of Research Opportunities", IEEE Internet of Things Journal, vol. 3, no. 6, December 2016.