INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

ORIGINAL ARTICLE

Enhanced Weighted Round Robin: A New Paradigm in Cloud Load Balancing

S. Sree Priya[1]*, Dr. T. Rajendran[2]

1 Research Scholar, PG and Research Department of Computer Science, Government Arts & Science College, Kangeyam-638108. (Affiliated to Bharathiar University, Coimbatore), Tamil Nadu, India
2 Assistant Professor, PG and Research Department of Computer Science, Government Arts & Science College, Kangeyam-638108. (Affiliated to Bharathiar University, Coimbatore), Tamil Nadu, India

OPEN ACCESS

Received: 08/12/2024
Accepted: 26/03/2025
Published: 06/05/2025

Citation: Sree Priya S, Rajendran DT (2025) Enhanced Weighted Round Robin: A New Paradigm in Cloud Load Balancing. Indian Journal of Science and Technology 18(15): 1220-1228. https://doi.org/10.17485/IJST/v18i15.3976

* **Corresponding author**.

sspriya.sivakumar@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives: The aim of this study is to suggest and recommend a load balancing algorithm based on enhanced weighted round robin (EWRR) technique to distribute incoming tasks or requests among various servers or resources in a cloud computing architecture. **Methods:** The study is accomplished by designing the enhanced weighted round robin (EWRR) load balancing and implemented in Cloud infrastructure by varying the number of jobs and virtual machines and evaluating the efficiency of the model. The experiments are simulated using CloudSim an open source tool and the effectiveness with regard to response time, idle time, task migration and delayed tasks are recorded. **Findings:** The obtained results are compared with round robin (RR) and weighted round robin (WRR) techniques and with the other state of art methods in the literature. The proposed EWRR performs better than all other approaches, such as Pragmatic Load Balancing (PLB) (28 ms) and Reinforcement Learning and Proficient Hybrid Lyrebird Falcon Optimization (RL-HFLO) (30 ms), with the shortest reaction time of 20 ms. Further, in comparison to greater values like 150 units in Round Robin and 120 units in K-means with RR, it reduces cumulative idle time to 75 units. Among all the methods examined, EWRR further reduces the number of delayed tasks to a mere 5. Even though EWRR involves 3 task migrations, which is slightly more than others, it results in better overall efficiency and utilization of resources. **Novelty:** The suggested EWRR model can be utilized as load balancing in cloud infrastructure in the varied cloud environments in a predictive way and enable the cloud service providers to make informed decisions. Dynamic parameters used by EWRR include each virtual machine's processing power, the quantity of incoming jobs, the duration of each job, and the VM's current burden (in real time). All of these EWRR settings produce intelligent scheduling that takes workload into account by preventing the blind assignment of jobs in RR and LC.

Keywords: Load Balancing; Weighted Round Robin; Enhanced Weighted Round Robin; Cloud Computing; Performance Evaluation

1 Introduction

Cloud Computing is a robust architecture that enables individuals and businesses to obtain services that are personalized to their own requirements. This approach includes numerous offers, including as solutions to storage problems, platforms for simple deployment, and easy accessibility of online services[1]. In cloud computing, load balancing distributes traffic and workloads such that no single server or computer is underutilized, over utilized, or idle. Load balancing improves total cloud performance by optimizing limited characteristics such as execution speed, response time, and overall stability.

Load balancing strategies spread the entire system's load across its numerous nodes to improve the utilization of resources and task response time. The key difficulties in load balancing involve geographically dispersed systems, single points of failure (SPoF), live instance migration, diverse devices, data storage device management, scale-out/scale-in, and algorithm complexity. Load balancing makes sure all nodes in the network reduce response times, minimizes expenses, and improves efficiency[2]. A load-balancing technique allocates user requests across the resources that are available, such as Virtual Machines (VMs). The load balancer in the cloud environment is shown in Figure 1. If the resources are not utilized with the proper load balancing, the quality of service (QoS) will suffer.

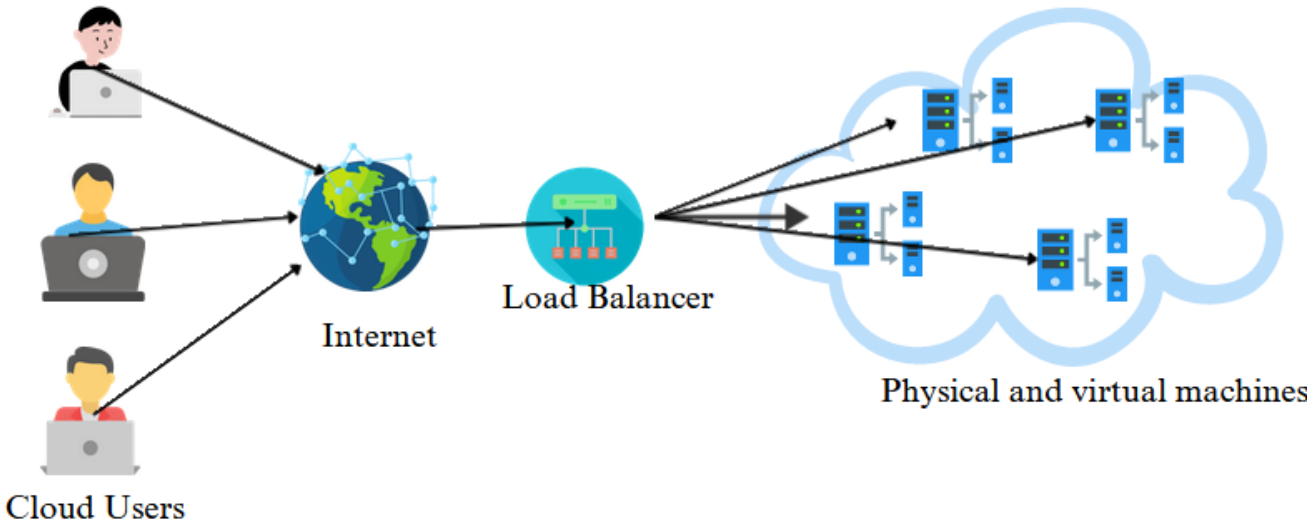

Fig 1. Load Balancing in Cloud Environment

Various strategies have been developed periodically for balancing the load on separate Virtual Machine (VM's) with the goal of improving the utilization of resources and to minimize down time. Generally, the load balancing algorithms are classified as static and dynamic. A static load balancing solution primarily avoids the present system environment. The environment contains data such as the loading situation. The system performance may be impacted by the VM's overload or underload. These approaches generally rely on the system's conventional behavior, transfer decisions and are independent of the actual present architectural state. Dynamic load balancing solutions locate the underutilized virtual machine in the system and then assign an appropriate huge amount to it. This approach assigns the task to all systems at an application tier[3]. Some of the recent research works carried out in the static and dynamic load balancing is as follows

The approach introduced in[4] by Mathanraj et.al employs principal component gradient round robin load balancing (PCGRLB) approach for balancing workloads in cloud servers in order to handle large workloads in a short period of time. Sharma proposes a novel technique called Pragmatic Load Balancing (PLB) to maximize reaction time, execution time, and cost[5]. The RR and least-connection algorithms (LC) are compared by Ma'arifah et.al in[6] to evaluate the performance of the Zevenet load balancer (ZLB). The QoS parameters findings demonstrate that the ZLB with the round-robin (RR) technique performs better and uses less CPU. It is concluded that employing the RR method to construct the ZLB will solve the issue of data traffic load sharing and reduce downtime of the server.

Elaskaan et.al examines the use of K-Means clustering technique to cluster servers inside the datacenter according to the similar usage rates. A RR approach is then used to assign task groups progressively to under-loaded clusters, and inside each cluster, a genetic algorithm allocates work to servers in the most optimal way[7].

S.M. Shetty et.al., suggested Modified Central Load Balancer (MCLB) technique to distribute the workload to each of the accessible VM's, preventing overloading and underloading. The MCLB algorithm is compared with the RR, Throttled and Equally Spread Current Execution Load algorithms[8]. Developing and operating a MCLB, particularly in big and dynamic contexts, can be challenging. This complexity frequently necessitates specialist knowledge and might raise software, equipment, and human expenses for installation and periodic repair. Singhal et al. designed a load balancing system using the Rock Hyrax, a metaheuristic approach which uses swarm optimization technique. The dynamic nature of cloud systems complicates load-balancing algorithms, demanding ongoing monitoring and correction. To meet the expanding needs of modern applications, cloud networks must use new load-balancing algorithms due to their increasing scale and complexity[9].

The model suggested by Khan et.al., uses a dynamic clustering technique with computed loads to divide VMs into over-loaded and under-loaded groups. To enhance the efficiency of clustering, the method combines Reinforcement Learning (RL) and a proficient Hybrid Lyrebird Falcon Optimization (HLFO) algorithm. HLFO combines the Lion Optimization Algorithm (LOA) and Falcon Optimization Algorithm (FOA) to increase the effectiveness of load balancing.[10]

A three phase technique named Regional Awareness Dynamic Scheduling Algorithm (RASA) was introduced by Khaleel et.al. Initially, a task classification model is used to categorize jobs based on essential parameters like CPU usage, storage utilization, and I/O operations. The nodes are then clustered using a novel merge-and-split-based coalitional game-theoretic procedure. In the third step, task placement is improved by adopting an Improved Sparrow Search Algorithm (ISSA) to address the classic SSA's delayed convergence and local optimization concerns. Moreover, it improves resource accessibility by 22% and efficiency by 27% and increases system throughput by 32%[11]. Table 1 lists the contributions of authors in the related study.

Table 1. Review of Existing Studies related to Load balancing

Author / Year	Method Used	Strengths	Limitations
Mathanraj et al., 2024[4]	Principal Component Gradient Round Robin (PCGRLB)	Handles large workloads quickly, balances cloud server load efficiently	Focused primarily on speed, may not fully optimize resource utilization
Sharma et al., 2024[5]	Pragmatic Load Balancing (PLB)	Improves reaction time, execution time, and cost	No mention of scalability or dynamic workload adaptability
Ma'arifah et al., 2024[6]	RR and Least-Connection (LC) with Zevenet Load Balancer (ZLB)	Reduces CPU usage, improves QoS, reduces server downtime	Limited to RR and LC techniques, may not handle highly dynamic loads
Elaskaan et al., 2024[7]	K-Means clustering + RR + Genetic Algorithm	Optimal workload allocation inside clusters, balances under-loaded clusters	Added complexity with multi-stage approach, potentially high overhead
S.M. Shetty et al., 2024[8]	Modified Central Load Balancer (MCLB)	Prevents VM overloading and underloading	High implementation cost and complexity in dynamic environments
Singhal et al., 2024[9]	Rock Hyrax (Swarm Optimization)	Adaptive to dynamic cloud environments, metaheuristic optimization	Requires continuous monitoring and tuning
Khan et al., 2024[10]	RL + Hybrid Lyrebird Falcon Optimization (HLFO)	Efficient dynamic clustering, hybrid optimization approach	Complex integration of RL and HLFO, increased computational overhead
Khaleel et al., 2024[11]	RASA (RADS + Game Theory + ISSA)	Improves resource accessibility (22%), efficiency (27%), throughput (32%)	Multi-phase model increases algorithmic complexity
Sinha et al., (2020),[12]	Round Robin (RR)	Simple to implement, evenly distributes load among available VMs	Ignores VM capacity and task characteristics, leading to possible VM overloading

Numerous methods now in use (such as RR, LC, and MCLB) are either flexible in highly dynamic settings or have significant processing cost when applied to large-scale systems. Furthermore, the dynamic load balancing techniques have some drawbacks like increased overhead, complexity, unpredictability in distributing the loads among the VM's. Static load balancing techniques have a predetermined load scheduling that determines the amount of load that can be distributed across different systems. It is intended for systems with minimal fluctuation in incoming load. In this load balancing method, traffic is distributed equally among the servers.

The literature study reveals that the most obvious benefit of RR load distribution is its ease of understanding and implementation. However, the ease of the RR algorithm is also its major weakness and most of the load balancers use weighted

RR (WRR) or more complicated algorithms based on RR. Furthermore, an effective load-balancing strategy should optimize resource use in Virtual Machines (VMs) to provide optimal user satisfaction and system efficiency. The suggested EWRR uses dynamic characteristics such as the processing power of each VM, the number of incoming jobs, the length of each job, and the present load of the VM (in real time). By avoiding the blind allocation of jobs in RR and LC, all of these EWRR parameters result in intelligent scheduling that considers workload. The objectives of the study are as follows,

- To suggest a load balancing algorithm with flexibility, scalability and ability to quickly handle system modifications.
- To assess the performance of the suggested load balancing algorithm.
- To recommend a load balancing model with optimal resource utilization and system efficiency.

2 Methodology

The general framework for the load balancing algorithms is shown Figure 2.

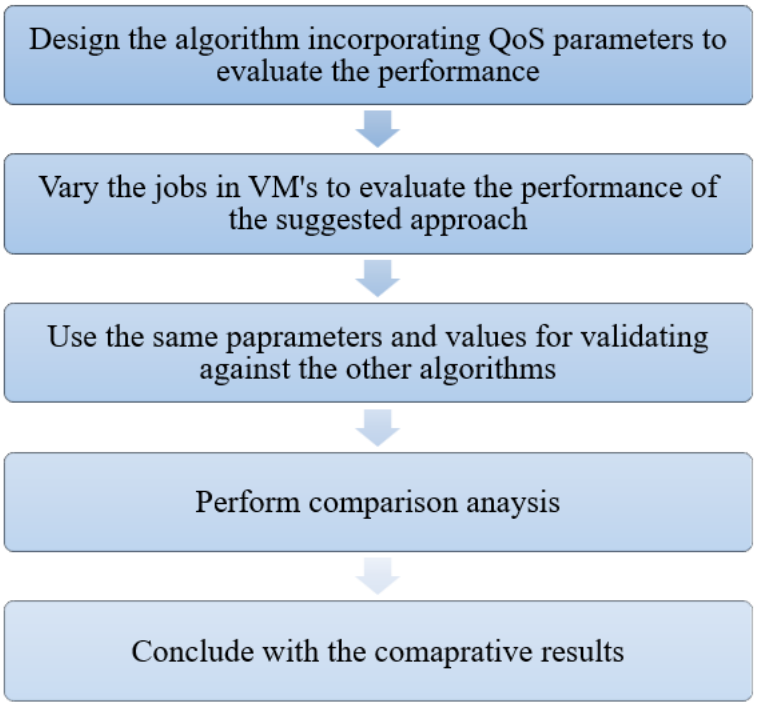

Fig 2. General Framework for Load balancing algorithm

The suggested model is designed with the QoS parameters like CPU usage, memory usage, makespan, response time and throughput. Then the tasks are assigned to VM'S using the suggested strategy and the efficiency is measured. Then the same values are used to run the other algorithms for comparison. The efficiency of all the algorithms used for comparison is assessed to draw conclusions about the outperforming approaches.

Round Robin (RR) Load Balancing: RR method serves as one of the widely used load balancing methods. RR is a method of distributing resources in cloud computing where each user takes turns using them in a predetermined order. Despite the emergence of other complex load balancing technologies, round robin remains significant because it is simple to comprehend and implement. It is the most straightforward approach for distributing traffic among a collection of servers. The load balancer forwards requests that come in to dedicated servers in a consecutive order. This load balancing mechanism is not adaptive because it fails to distribute incoming requests based on the server's current workload. For instance, if server A is currently processing a huge request, the load balancer continues to send the following request if it is its turn. During the same time, other machines might stay passive after completing their assigned responsibilities. Ideally, the load balancer should deliver the request to the server that is currently idle, however it does not. Johanna et al. observed that the algorithm with the best responses in static techniques is Round Robin [13].

RR load balancing is a load sharing approach that distributes client demands over a group of servers to effectively balance server load. It performs best when the servers have similar storage and computing resources. RR network load balancing distributes requests for connections to web servers in the sequence in which they are delivered. Consider a group that has three servers: A, B, and C.

- Server A receives the 1st request
- Server B receives the 2nd request
- Server C gets the 3rd request

This order causes the load balancer to continue sending requests to servers. This distributes server load uniformly, allowing the balancer to handle significant amounts of traffic. It is a sequential resource distribution that uses the round-robin method to distribute jobs. First, it selects the first node at random and allocates work to the other nodes in a round robin fashion. Processors are assigned to each process in a cyclical fashion, with no priority specified. It leads to a speedier response when the workloads are distributed similarly among the processes [14].

Weighted Round Robin (WRR) Load Balancing: The WRR server load balancer is a static load balancing since it distributes incoming traffic without changing the state of the servers. This algorithm is meant to function with predetermined weights and jobs that are dispersed based on the weight values. Processors with stronger capability are assigned a higher value. The highest-weighted servers will receive more work. When all weights reach the same level, servers will experience consistent traffic [14]. For instance, the server with the most processing power, say ServerA, will be given the highest weight. It will also receive the highest percentage of inbound requests from the load balancer. A server, say ServerB, having half the processing capacity of ServerA will be assigned a weight equal to half of ServerA's real weight. In addition, it will receive the appropriate proportion of requests that come from the load balancer. A server, say ServerC, with the lowest parameters will be allocated the least amount of weight and get the smallest fractions of requests coming in from the load balancer.

Enhanced Weighted Round Robin (EWRR) Load Balancing: The proposed EWRR algorithm is the most efficient method for allocating work to the most appropriate virtual machines. To select which VM should be assigned to each job, this algorithm takes into account the VMs' processing capability, number of incoming jobs and the length of the job. Further data, such as the VM's current workload, is used for assigning tasks to the appropriate VMs in this algorithm's dynamic scheduling (at runtime). Because of the detailed run-time data, it is possible that the job will run for a longer time to complete than the initial computation because of the execution of extra cycles using the same instructions.

EWRR load balancer saves the scheduling controller by transferring waiting work from one of the densely packed VMs to an idle slot in another underutilized virtual machine. When a job is executed in one of the VMs, the load balancer employs a resource probe to identify underutilized VMs. Load balancing will not cause any job transfer if there are no unutilized VMs. If a job is discovered to be overcrowded, it might be relocated to an underutilized or underutilized virtual machine. When any of the VM obligations are performed, the load balancer checks the resource load. To avoid putting a strain on the virtual computers, it never assesses the resource load independently.

3 Results and Discussion

CloudSim, an open source framework is used to simulate the load balancers suggested in this study. The response time, number of work migrations, idle time for all tasks, and number of delayed tasks under heterogeneous and homogeneous conditions are evaluated for the suggested approach and it is represented graphically. It is validated with the simple round robin and weighted round robin approaches and the results of the performances are compared to draw a conclusion about the outperforming load balancer. The study took into account the techniques based on round robin for validating against the suggested model to avoid major deviation from the results and also to recommend enrichment over the existing traditional models.

Response time: The response time is the entire time taken by the server to process and respond to incoming requests. In selecting the optimal server, the least response time method considers both server response time and connections that are active. Load balancers employ this method to provide faster service to all users. The least response time load balancing technique considers both the total number of active connections on each server and the average response time. The tasks response time

is computed using two components as follows

$$\begin{aligned} &\text{Response time} \\ &= \\ &\text{Task Ready time} + \text{processing time} \\ &R_i \\ &= \\ &R(trt) + R(proc) \end{aligned} \tag{1}$$

Where Ri is the response time of task T_i, R(trt) is the task ready time of Task Ti and R(proc) is the processing time of task T_i.

The R(trt) shows the maximum time required for the task Ti's data to reach the server, which is specified as follows

$$R(trt) = \max\ R(ntrt) \tag{2}$$

R(ntrt) represents the time required for the needed information to be sent between server s1 to server s2, where s2 is the server wherein the job Ti will be processed based on the scheduling configuration Xi, and s1 is the server in which the task's primary task Ti is processed.

The processing time R(proc) is defined as the time taken for the server n1 to process the Ti based on the scheduling configuration Xi is computed as follows

$$R(proc) = Ti(size)/n1(freq) \tag{3}$$

Where Ti(size) denotes the required CPU cycles for tasks Ti and n1(freq) indicates the CPU frequency of server n1.[15]

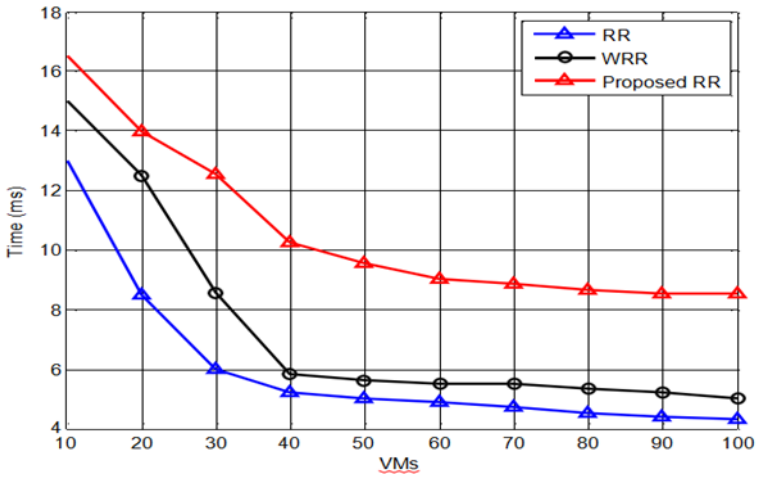

Fig 3. Response time in RR, WRR and EWRR load balancers

The time vs VM's graph is drawn, and it is depicted in Figure 3 shows the varying processing capabilities of the VM's. It is evident from Figure 3 that the proposed ERR outperforms the other two load balancing algorithms in terms of response time in heterogeneous and homogeneous environments (RR and WRR). As a result, more tasks in heterogeneous settings will be assigned to VMs with more capacity. This permits the project to be completed in a shorter period of time.

Task Migration: Task migration is defined as the movement of partially completed work to a different node in a distributed system. It can occur for various reasons, including load balancing, hardware maintenance, and responding to errors. The load balancer selects whether to migrate a workload from a strongly loaded VM to an idle VM or the least loaded VM. The load on each node is automatically adjusted, and tasks are migrated based on established constraints, reducing network cost[16], [17]. Figure 4 shows the number of tasks migrated from one VM to another due to the load. The adjustment of load is found to be more in the case of ERR than the other two methods.

The proposed EWRR model performs tasks more consistently than RR and WRR when the number of VMs increases, as illustrated in Figure 4. With 40 VMs, EWRR can handle up to 13 tasks, whereas RR and WRR both show a low and flat trend with fewer job handling. Workload saturation results in all algorithms dropping to zero jobs for more than 60 VMs. Overall, the proposed EWRR exhibits excellent task management and flexibility, particularly in setups for mid-level VMs.

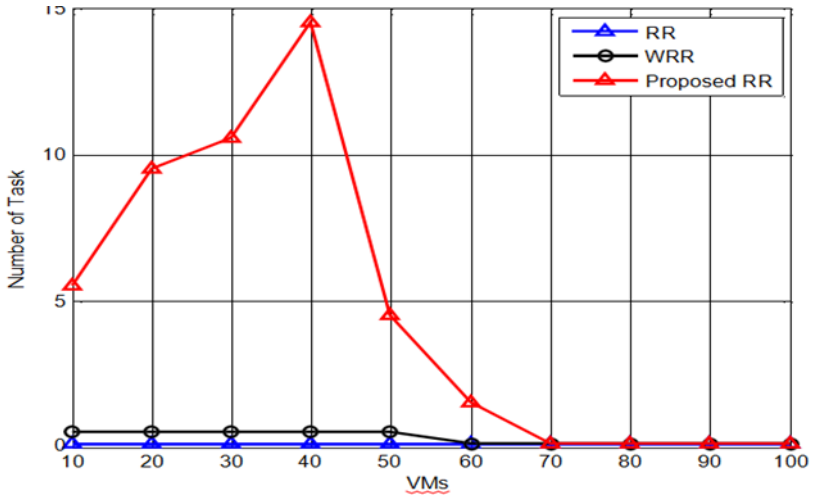

Fig 4. Number of tasks Vs VM's to show the task migration

Cumulative idle time: Despite being idle, VMs continue to operate. CPU, memory, and storage resources that can be used by active PCs are consumed by idle virtual machines[18]. The load balancer allocates the work area in a way that minimizes the aforementioned features in order to cut down on idle time. As seen in Figure 5, the EWRR model gradually decreases idle time in contrast to the RR and WRR models. It is later shown that, in comparison to EWRR, RR and WRR have more idle VMs. Excellent task management and flexibility are demonstrated by the suggested EWRR, especially in mid-level VM configurations.

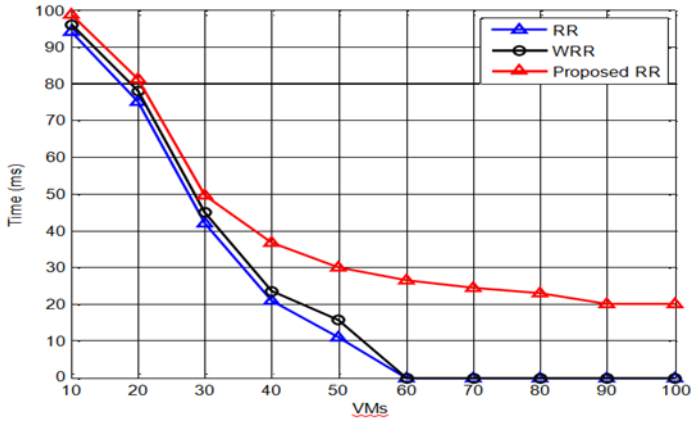

Fig 5. Time Vs VM's (Cumulative idle time)

Figure 5 demonstrates that response times for all algorithms decrease with an increase in VMs. When VMs surpass 60, RR and WRR achieve nearly nil response times, but the suggested EWRR keeps a steady response time of about 20 ms after 50 VMs.

Delayed Tasks: Figure 6 illustrates that the proposed EWRR approach can manage more number of tasks.

The EWRR model shows improved outcomes for the number of delayed tasks, which is caused by higher workload allocation in VMs with greater capacity[19]. It starts with about 1450 jobs with 10 VMs and stabilizes at about 850 tasks even when the number of VMs reaches 100. Given that it more effectively maintains workload balance and resource utilization across expanding cloud architecture; this demonstrates EWRR's better scalability and adaptability. EWRR's capacity to dynamically adjust to runtime conditions and distribute workloads based on real-time VM performance measurements makes it the most successful of the three methods. The slow reduction in EWRR's task count is negligible in comparison to the sharp dips in WRR.

Table 2 provides the comparative analysis with some more existing methods rather than restricting the analysis with round robin based methods. It shows the performance measures like response time, task migration, cumulative idle time and number

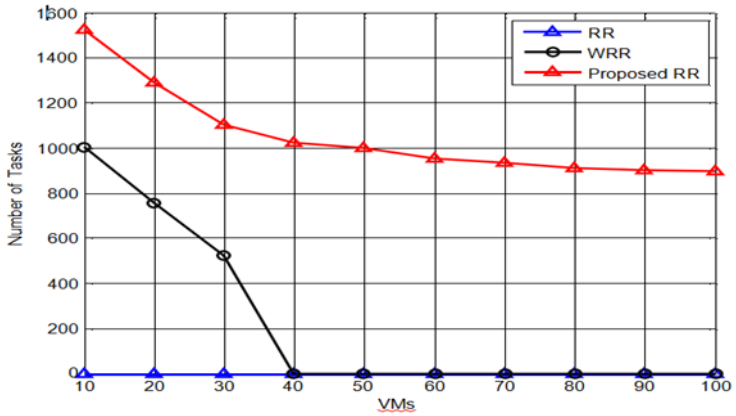

Fig 6. Number of delayed tasks Vs VM's

of delayed tasks for the existing and proposed methods. The response time for the proposed method is found to be lower than the other existing methods. Task migration is critical for maximizing resource efficiency, increasing reaction time, and reducing delays, particularly in dynamic and diverse contexts. It is found to be high for the proposed round robin method.

Table 2. Comparison Analysis of Existing methods and Proposed Methods

Author/Year	Method	Response time (ms)	Task Migration	Cumulative idle time	Number of delayed tasks
Sharma et al.,2024 [5]	PLB	28	2	80	7
Ma'arifah et.al, 2024 [6]	RR and LC RR with ZLB	25	1	90	8
Elaskaan et al., 2024 [7]	K-means and RR	33	1	120	7
Singhal et al., 2024 [9]	Rock Hyrax	35	2	110	6
Khan et al., 2024 [10]	RL and HFLO	30	1	95	9
Toofani et al. [20]	Round Robin	38	2	150	11
Proposed Study -EWRR	Enhanced Weighted Round robin	20	3	75	5

The comparison analysis shows that the proposed EWRR algorithm performs better than existing methods on certain performance metrics. With the lowest response time of 20 ms, EWRR beats methods such as PLB (28 ms), K-Means with RR (33 ms), and Rock Hyrax (35 ms) Furthermore, EWRR outperforms other strategies like RR with ZLB (90 units) and Weighted Round Robin (100 units) in terms of resource utilisation, achieving the lowest total idle time of 75 units. Additionally, compared to other approaches, where delayed jobs range from 6 to 11, EWRR has the lowest number of delayed tasks (5), indicating greater task handling efficiency. The task migration count (3) of EWRR is slightly higher than that of other methods such as RR, LC, and RL-HFLO; however, this minor increase is compensated for by a significant decrease in idle time and work that is put off. Compared to current load balancing algorithms, EWRR offers improved load distribution, reduced latency, and increased system responsiveness; in general, the trade-offs are well-balanced.

4 Conclusion

A load balancing algorithm based on enhanced weighted round robin (EWRR) is presented in this work with the goal of enhancing the traditional RR mechanism. Reduced response time, less task migration, less cumulative idle time, and less delayed tasks are just a few of the important performance indicators that show how efficient the suggested EWRR is. EWRR is unique in that it optimizes task allocation in cloud environments by dynamically integrating real-time resource characteristics, including processing capacity, current workloads, and job durations. The exact assignment of weights to VMs, which is more difficult in

large-scale cloud infrastructures, presents difficulties for the method. Additionally, the EWRR method's flexibility for varied workloads may be limited by its potential for suboptimal performance under highly changeable service time situations. Future research will need to focus on smart and adaptive weight-transfer mechanisms, with possible implementation based on machine learning and artificial intelligence. To figure out what of the alternatives suit some cloud arrangements and applications, a contrast of static load balancing techniques and dynamic load balancing techniques is also suggested. Ultimately, in order to develop more reliable and efficient algorithms that would be adapted to the varying and dynamic nature of cloud environments nowadays, testing the EWRR and other load balancing techniques with real, dynamic workload conditions will be needed.

References

1) Prity FS, Hossain MM. A comprehensive examination of load balancing algorithms in cloud environments: a systematic literature review, comparative analysis, taxonomy, open challenges, and future trends. *Iran Journal of Computing Science*. 2024. https://doi.org/10.1007/s42044-024-00183-y.
2) Ajil A, Kumar S. Categories of LB technique in cloud infrastructure. *AIP Conference Proceedings*. 2024;2742(1). https://doi.org/10.1063/5.0184662.
3) Zhou J, Lilhore UK, M P. Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing. *Journal of Cloud Computing*. 2023.
4) Mathanraj E, Reddy RN. Enhanced principal component gradient round-robin load balancing in cloud computing. *The Scientific Temper*. 2024;15(1):1806–1815. https://doi.org/10.58414/SCIENTIFICTEMPER.2024.15.1.32.
5) Sharma T, Bedi RP. Design and Development of Pragmatic Load Balancing Algorithm for Cloud Environment. *Wireless Personal Communications*. 2024 May;p. 1–21. https://doi.org/10.1007/s11277-024-11117-z.
6) Ma'arifah W, Sarmini S. Performance Comparison of Zevenet Multi Service Load Balancing with Least Connection and Round Robin Algorithm. *JOIV: International Journal on Informatics Visualization*. 2024;8(2):882–890. https://dx.doi.org/10.62527/joiv.8.2.1985.
7) Elsakaan N, Amroun K. A novel multi-level hybrid load balancing and tasks scheduling algorithm for cloud computing environment. *Journal of Supercomputing*. 2024;80:13434–13474. https://doi.org/10.1007/s11227-024-05990-5.
8) Shetty SM, Shetty S. Analysis of load balancing in cloud data centers. *Journal of Ambient Intelligence and Humanized Computing*. 2024;15:973–981. https://doi.org/10.1007/s12652-018-1106-7.
9) Singhal S, Sharma A, Verma PK, Kumar M, Verma S, Kaur M, et al. Energy Efficient Load Balancing Algorithm for Cloud Computing Using Rock Hyrax Optimization. *IEEE Access*. 2024 Mar. https://doi.org/10.1109/ACCESS.2024.3380159.
10) Khan AR. Dynamic Load Balancing in Cloud Computing: Optimized RL-Based Clustering with Multi-Objective Optimized Task Scheduling. *Processes*. 2024;12(3):519. https://doi.org/10.3390/pr12030519.
11) Khaleel MI. Region-aware dynamic job scheduling and resource efficiency for load balancing based on adaptive chaotic sparrow search optimization and coalitional game in cloud computing environments. *Journal of Network and Computer Applications*. 2024;221:103788. https://doi.org/10.1016/j.jnca.2023.103788.
12) Sinha G, Sinha DK. Enhanced Weighted Round Robin Algorithm to Balance the Load for Effective Utilization of Resource in Cloud Environment. *EAI Endorsed Transactions on Cloud Systems*. 2020;6(18). http://dx.doi.org/10.4108/eai.7-9-2020.166284.
13) Johanna CVM, Veliz MR, García LC. Static and dynamic load balancing algorithms most commonly used in different environments. *AIP Conference Proceedings*. 2024;2994(1). https://doi.org/10.1063/5.0187582.
14) Vijay R, Sree TR. Resource Scheduling and Load Balancing Algorithms in Cloud Computing. *Procedia Computer Science*. 2023;230:326–336. https://doi.org/10.1016/j.procs.2023.100326.
15) Wang Z, Goudarzi M, Gong M, Buyya R. Deep reinforcement learning-based scheduling for optimizing system load and response time in edge and fog computing environments. *Future Generation Computer Systems*. 2024;152:55–69. https://doi.org/10.1016/j.future.2023.10.012.
16) He T, Buyya R. A taxonomy of live migration management in cloud computing. *ACM Computing Surveys*. 2023;56(3):1–3. https://doi.org/10.1145/3615353.
17) Zhu X, Yao W, Wang W. Load-aware task migration algorithm toward adaptive load balancing in Edge Computing. *Future Generation Computer Systems*. 2024 Aug;157:303–312. https://doi.org/10.1016/j.future.2024.03.014.
18) Joshi NA. Rational load balancing in collaborated cloud computing environments. *Engineering Research Express*. 2024 May;6(2):025008. https://doi.org/10.1088/2631-8695/ad4233.
19) Shamsa Z, Rezaee A, Adabi S, Rahimabadi AM, Rahmani AM. A distributed load balancing method for IoT/Fog/Cloud environments with volatile resource support. *Cluster Computing*. 2024 May;p. 1–40. https://doi.org/10.1007/s10586-024-04403-9.
20) Toofani AK, Kumar A, Giri AK, Verma A, Kumar N. Energy Efficiency and Load Balancing Algorithm for Cloud Environment. *Smart City Insights*. 2024;1(1):7–12. Available from: https://www.sci.reapress.com/journal/article/view/17.