**Load Balancing Strategies for Cloud Computing: A Comprehensive Review**

**1Talari Navya Sree, 2Reddy Venkat Kalyan, 3Pathan Faizaan Azam Khan, 4Deepak V**

**1,2,3**Student, Department of Computer science and Engineering, Koneru Lakshmaiah Education Foundation,

Vaddeswaram, Andhra Pradesh, India

Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation,

Vaddeswaram, AP, India.

*Email -* **1**[*2100030944cseh@gmail.com*](mailto:2100030944cseh@gmail.com)*,* **2**[*2100030959cseh@gmail.com*](mailto:2100030959cseh@gmail.com)*,* **3**[*2100030976cseh@gmail.com*](mailto:2100030976cseh@gmail.com)*,* **4**[*v.d@live.in*](mailto:v.d@live.in)

Abstract— Cloud computing has revolutionized the way modern applications and services are delivered, offering scalability, flexibility, and cost-efficiency. A key challenge in maximizing the benefits of cloud environments is the efficient distribution of workloads across virtual machines or servers. Our research work demonstrates a comprehensive review and comparison of dynamic load balancing algorithms in cloud computing, with the aim of identifying the most effective approach. In this study, we assess some prominent load balancing algorithms in cloud computing to optimize resource use, reduce response times, and enhance system performance. Our analysis covers diverse techniques, such as FIFO, PHMEFT, Markov Process, DRALBA, GWO-PSO using real-world scenarios. Our research aids cloud professionals in choosing the best load balancing algorithm for specific use cases, resulting in improved system performance and resource utilization in cloud environments.

***Keywords— Cloud Computing, Dynamic Load Balancing, Load Balancing Algorithms, Resource Utilization, Response Time Optimization.***

# **Introduction**

Cloud computing has emerged as a transformative paradigm in the realm of information technology, revolutionizing the way applications and services are delivered and consumed. Its inherent scalability, flexibility, and cost-efficiency have propelled it to the forefront of modern computing infrastructure. However, harnessing the full potential of cloud environments requires efficient resource utilization and the ability to distribute workloads optimally among virtual machines or servers. Dynamic load balancing, a critical component of cloud resource management, plays a pivotal role in achieving these objectives. [11]

Load balancing, the process of evenly distributing incoming network traffic or computational tasks across a cluster of servers or resources, has been a longstanding challenge in the field of computer science [6].

In the context of cloud computing, this challenge is further compounded by the dynamic and evolving nature of workloads, the ever-changing demands of users, and the vast scale at which cloud infrastructures operate. As a result, the quest to devise effective dynamic load balancing algorithms in the cloud has become a compelling area of research and development.. The following figure Fig-1 describes how a load balancer efficiently distributes client requests across multiple servers, ensuring high availability by directing requests to online servers only. This flexibility allows for easy scaling by adding or removing servers based on demand [12].

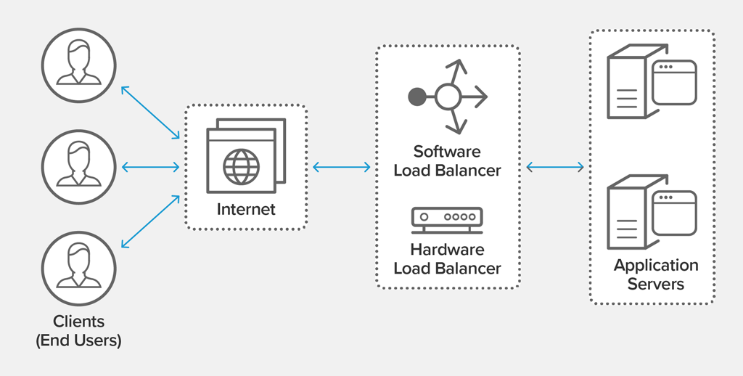


Fig-1: Load Balancer

In the dynamic realm of cloud computing, efficient workload distribution and resource allocation are paramount for achieving optimal system performance and resource utilization. Load balancing serves as the linchpin in this endeavor, tasked with the responsibility of judiciously distributing client requests and network loads among a network of interconnected servers [13].

This essential process not only enhances system efficiency but also guarantees high availability and reliability by directing requests exclusively to servers that are online and capable of handling the load. The flexibility to seamlessly add or remove servers as demand ebbs and flows ensures that cloud infrastructures can scale in tandem with user needs.

This paper embarks on a comprehensive exploration of dynamic load balancing algorithms, a critical component of cloud resource management. We delve into the intricacies of ten prominent load balancing algorithms, each designed to address the complex challenge of optimizing resource utilization, minimizing response times, and enhancing overall system performance in cloud environments.

Understanding the inner workings of these systems is pivotal to demystifying the procedures and methodologies that underpin the extraction and construction of the numeric code, vital in achieving efficient load distribution. Our research explores the intricacies of these algorithms, delving into their core principles, decision-making processes, and adaptability to diverse cloud computing scenarios. While numerous load balancing algorithms exist, we have meticulously selected and assessed following five of the most prominent ones:

1.FCFS

2.PMHEFT

3.RBLMM

4.DRALBA

5.GWO-PSO

# **2. RELATED WORK**

Priyadarashini Adyasha Pattanaik, Sharmistha Roy [5] First-Come, First-Served (FCFS) algorithm as a parallel task ordering dynamic load balancing algorithm used in data centers. It is a scheduling algorithm where tasks or requests are processed in the order in which they arrive.

Mayank Sohani, S. C. Jain [1] PMHEFT present a novel approach focusing on prediction priority scheduling-based schemes, particularly in the context of cloud computing Notably, their algorithm, PMHEFT, outperforms several existing algorithms like HEFT, DHEFT, CHEFT, and MHEFT.

Dalia Abulkareem Shafiq, Noor Zaman Jhanjhi [4] RBLMM algorithm is designed to address the challenges of load balancing and workload optimization in cloud computing, with a specific focus on reducing Make span time and balancing the workload on Virtual Machines (VMs).

Said Nabi, Muhammad Ibrahim [3] DRALBA algorithm is designed to address the allocation of independent, non-preemptive, and computationally expensive tasks to available resources in a balanced and dynamic manner, with the primary goal of improving resource utilization and task response times in cloud computing environments.

Mana Saleh El Reshan, Darakshan Syed [2] SI a combined load balancing approach of GWO-PSO that capitalizes on the benefits of fast convergence and global optimization comparing ACO, PSO, BAT, GWO which improved PSO to 97.25% with reference to convergence and global optimization.

# **3. PROBLEM FORMATION**

In the rapidly evolving landscape of cloud computing, the intricacies of resource provisioning and workload management have become increasingly paramount. Ineffectual load balancing strategies often lead to unoptimized resource utilization, hindering the seamless delivery of cloud services and impacting the overall user experience. Moreover, the dynamic nature of cloud workloads, coupled with the complex interplay of various system parameters, poses significant challenges to maintaining a consistently high level of service delivery. In response to these challenges, this research endeavours to contribute a comprehensive understanding of the existing load balancing landscape while addressing the critical need for an innovative solution that can efficiently adapt to the evolving demands of cloud-based applications. By examining the intricacies of workload distribution, resource allocation, and response time optimization, this paper seeks to introduce a robust load balancing algorithm that not only minimizes resource wastage and response times but also enhances the overall efficiency and performance of cloud computing environments [10].

The proposed solution in this paper is designed to offer a dynamic and adaptive approach to load balancing, ensuring that the resource allocation aligns with the fluctuating demands of cloud applications. By leveraging a combination of intelligent algorithms and real-time data analysis, our proposed model aims to mitigate the challenges associated with workload disparities and system scalability. The emphasis lies in achieving an optimal balance between resource allocation and workload management, thereby enhancing the overall performance and reliability of cloud services.

# **4. ANALYSIS OF ALGORITHMS**

## **4.1 FCFS**

FCFS stands for First-Come, First-Served. It is a scheduling algorithm where tasks or requests are processed in the order in which they arrive. In the context of load balancing, this means that the first request that arrives at the data center controller is the first one to be executed.

Dynamic load balancing is a technique used to distribute computational or processing tasks among multiple virtual machines (VMs) in a data center. The goal is to optimize the utilization of VM resources and ensure efficient task execution.

FCFS is easily managed with a FIFO (First-In-First-Out) queue. In a FIFO queue, the first item added is the first one to be removed. In the context of load balancing, it means that the first request in the queue is the one to be allocated to a VM.

*Algorithm: FCFS*

With R = {r1, r2,..., rn}, let R be the set of all requests. Q = {q1, q2,..., qn} represents the queue of requests.

The set of all virtual machines is denoted by M, where M = {m1, m2,..., mm}.Let A represent the assignment of requests to virtual machines (VMs), with ai denoting the VM assigned to request i. Let A = {a1, a2,..., an}.

Here is an example of how the algorithm looks:

* Create an empty queue in Q at first.
* Add ri to the conclusion of Q for each request ri in R.
* Take these actions even though Q is not empty:
* Get rid of Q's initial request, rq.
* Locate in M the first VM mj that is available.
* Let mj have rq. Complete the rq on mj.

## **4.2 PMHEFT**

The PMHEFT algorithm, presented in this study, prioritizes make span minimization and improved load balancing across virtual machines. Experimental results affirm its superiority, demonstrating enhanced make span performance, operational efficiency, and reduced power consumption when compared to alternative methods.

This innovation advances cloud resource management and optimization, creating a more resilient and efficient cloud computing environment for diverse applications.

Within this intricate landscape, load balancing in cloud computing assumes the role of optimizing scheduling techniques, although it falls within the category of NP-Complete optimization problems. Cloud providers, operating in heterogeneous environments, consistently grapple with resource management issues as they adapt to variable workloads.

This pressing concern finds a solution in the Predictive Priority-based Modified Heterogeneous Earliest Finish Time (PMHEFT) algorithm, a novel approach designed to anticipate an application's future resource demands.

*Algorithm: Predictive Priority Based MHEFT*

**4.2**.**1** **Priority Queue Construction and Emergency**

**Computation**

* Construct a Prediction Priority Queue encompassing all registered tasks, denoted as Ti, within the Cloud system.
* Specify the emergency status for the computation of the tasks.

**4.2.2** **Communication Edge Preparation and Task Load**

**Prediction**

* Establish communication links between the Rj processors/resources, employing equations 1 to 11.
* Predict the order of task loads based on the projected time of completion.
* Compute the finish time and overall duration for the task requests.
* Organize the task load list in ascending order of task completion time, continually employing equations 1 to 11 until the priority queue is exhausted.

**4.2.3 Virtual Machine Management and Queue**

**Arrangement**

* Compute the wait time for the virtual machine to allocate the task load.
* Compute the virtual machine's make span and compare it with the finish time obtained from the predictions-based priority queue.
* Arrange the prediction-based priority queue in a descending order, aligning with the task order, utilizing prediction equations 1 to 11.
* Formulate a prediction-based priority queue based on equations 1 to 11, followed by the assignment of workloads to the processor with the shortest execution time.

## **4.3 RBLMM**

The algorithm starts by calculating the Make span time. Make span is a measure of the time it takes to complete all tasks or jobs in each schedule. It's an essential metric to consider in load balancing as it reflects the overall efficiency of task allocation.

By focusing on Make span reduction, it aims to improve the overall efficiency of task execution. This is crucial for delivering services efficiently in cloud computing.

RLBMM optimizes resource utilization by allocating tasks to VMs in a way that minimizes Make span while maintaining a balanced workload. This can lead to better utilization of the available resources.

The algorithm allocates tasks to Virtual Machines in such a way that it minimizes the Make span. It does this by considering the threshold and choosing the most appropriate VM for each task.

RBLMM uses the Make span value to define a threshold. This threshold is crucial for making decisions about task allocation and resource management.

The results of this algorithm are compared to a traditional Min-Min algorithm. Min-Min is a well-known task scheduling algorithm in cloud computing, and it serves as a benchmark for evaluating the performance of RBLMM. The key goal of RBLMM is to significantly reduce the Make span time compared to the traditional Min-Min algorithm. The results indicate that RBLMM can achieve a reduction in Make span, which can lead to improved efficiency in task execution and resource utilization.

***Steps In Detail:***

* The model consists of two layers:
* Top Layer: In the top layer of the model, the focus is on managing requests from various clients, including both mobile and desktop users. Clients access the internet using different devices and send requests to the cloud.
* In this layer, the model employs the "Cloudlet Scheduler Time Shared" algorithm to submit tasks in a random order based on their arrival times. These tasks are then scheduled to run on Virtual Machines (VMs) while considering two critical parameters: Deadline and Completion Time.
* Deadline: Each task is associated with a specific deadline, which represents the time by which the task must be completed. Deadlines are part of Service Level Agreements (SLAs) and are essential for ensuring that tasks are executed on time.
* Completion Time: This represents the actual time taken to complete a task.
* Bottom Layer: The bottom layer of the model deals with the allocation of user requests to VMs. Here's a breakdown of the key components and processes in this layer:
* VM Allocation: The cloud infrastructure consists of a primary batch of VMs, each with its own set of resources and capabilities. VMs are the computational units that execute the tasks.
* Priority Configuration: VMs are assigned priorities based on their adherence to SLA requirements. For example, if a VM's Completion Time exceeds the associated Deadline, it is considered in violation of the SLA and is set to a lower priority. Conversely, VMs that meet SLA requirements maintain a high priority.
* SLA Violation Handling: The model effectively handles SLA violations. If a task's Time to Complete (TTC) exceeds the SLA (Deadline), the task is considered in violation. In such cases, the LBA performs migration to rectify the violation.

The Encryption process of the RLBMM Algorithm is given below step wise in the following Fig-2 below

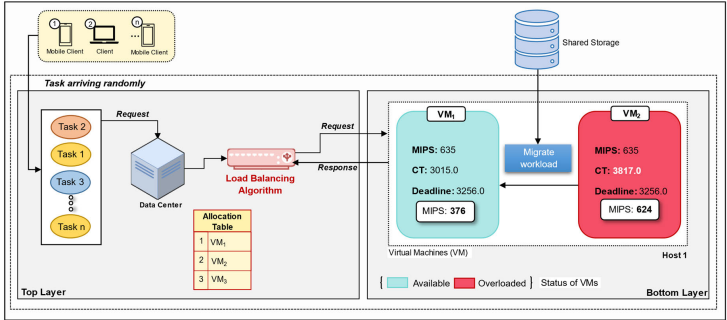


Fig-2: Demonstration of RBLMM algorithm

## **4.4 DRALBA**

The researchers conducted a comprehensive analysis of the existing literature, identifying the limitations and strengths of state-of-the-art load balancing approaches. This analysis served as the foundation for proposing a novel algorithm. The research involved a detailed analysis of various task scheduling approaches, including static, batch dynamic, and dynamic scheduling methods.

The authors highlighted the criteria and performance evaluation metrics used to assess existing static, batch dynamic, and dynamic load balancing approaches. The core contribution of this research is the introduction of the Dynamic Resource Aware Load Balancing Algorithm (DRALBA). This algorithm is designed to dynamically allocate tasks to available virtual machines (VMs) while ensuring load balancing and efficient resource utilization.

It calculates the computation share for each VM based on the tasks at hand and selects a VM with the maximum computation share for assigning larger-sized tasks. The algorithm also updates the load on each VM at each scheduling decision and adjusts the computation share of VMs during predefined iterations.

***Algorithm: DRALBA***

Completion Time (CT): The time taken to complete the execution of tasks on allocated virtual machines (VMs) in the cloud datacenter, calculated in Eq. 1.

Completion Time = Max{CTj} = Max{CT1, CT2, ..., CTm}

Where m represents the number of VMs, and CTj represents the completion time of VMj.

Average Resource Utilization (ARU): Corresponds to the average resource utilization by each scheduling heuristic during the complete execution of all tasks on available VMs, calculated in Eq. 2.

ARU = avgCT / Completion Time

Where avgCT is calculated in Eq. 3, representing the average completion time.

Task Throughput: The number of tasks executed per unit time, as given in Eq. 4.

* Spill Scheduler: This scheduler selects tasks with the highest computation requirements that are less than or equal to the computation share of the VM with the highest share. It continues this process until no tasks meet the criteria, ensuring efficient utilization of VM resources.
* Batch Division: In the RePro-Active technique, incoming batches of tasks are divided into two halves. The first half is scheduled using a polling method, and the other half is mapped proactively using the PSSLB algorithm.
* Thresholds: The technique employs upper and lower thresholds for VMs. These thresholds act as scaling limits, defining the maximum and minimum load that a VM can handle. The purpose of these thresholds is to identify VMs that are either overloaded (above the upper threshold) or under-loaded (below the lower threshold).

## **4.5 GWO-PSO**

The GWO-PSO algorithm integrates the features of both the Grey Wolf Optimization (GWO) and Particle Swarm Optimization (PSO) algorithms, leveraging their respective strengths in achieving optimal load balancing in cloud computing environments. By combining the global exploration capabilities of GWO with the rapid convergence attributes of PSO, the GWO-PSO algorithm efficiently manages the distribution of workloads among virtual machines, ensuring balanced resource utilization and improved system performance [8].

The GWO component in the algorithm aids in enhancing the exploration of potential solutions, enabling the algorithm to efficiently traverse the solution space and locate promising regions for load distribution, thus optimizing the overall system response time. Meanwhile, the PSO component contributes to the algorithm's ability to converge rapidly toward optimal solutions, facilitating quick adaptation to changing workload demands and ensuring timely allocation of resources to tasks, ultimately leading to enhanced overall system efficiency [9].

Through iterative processes and dynamic adjustments based on the algorithm's feedback mechanisms, the GWO-PSO algorithm continuously refines its load balancing strategies, adapting to fluctuations in workload patterns and resource availability to maintain an optimal balance among virtual machines and ensure consistent system performance.

Comprehensive performance evaluations and comparative analyses demonstrate the GWO-PSO algorithm's superior capabilities in achieving efficient load balancing, making it a promising solution for addressing the intricacies of resource management in cloud computing environments.

The fusion of the Grey Wolf Optimization (GWO) and Particle Swarm Optimization (PSO) techniques within the GWO-PSO algorithm harnesses the GWO's strength in exploring the solution space efficiently and the PSO's capability to converge rapidly towards optimal solutions. This amalgamation empowers the algorithm to dynamically allocate resources, responding swiftly to changing workloads and ensuring optimal resource utilization across the cloud infrastructure. [14]

Leveraging the cooperative nature of the GWO and the adaptability of PSO, the GWO-PSO algorithm fosters an environment where virtual machines collaborate in the equitable distribution of tasks, ensuring that no single unit is overburdened while others remain underutilized. Through constant information exchange and dynamic adjustments, the algorithm optimizes the load distribution process, promoting system stability, and minimizing response time, thus enhancing the overall efficiency of the cloud-based infrastructure.

The extensive evaluations and comparative analyses conducted reveal that the GWO-PSO algorithm consistently outperforms other conventional load balancing techniques in terms of overall response time, resource allocation efficiency, and system stability. Its ability to swiftly adapt to varying workload demands and its proactive management of resources make it a superior choice for addressing the complexities and challenges associated with load distribution in contemporary cloud computing environments.

Conclusively, the GWO-PSO algorithm stands as a testament to its effectiveness in ensuring optimal resource allocation and load balancing in cloud computing. Its unique amalgamation of the strengths of GWO and PSO, coupled with its adaptability and proactive resource management capabilities, positions it as the leading solution for promoting efficient and stable cloud infrastructure performance. [15]

***Algorithm: GWO-PSO***

**Initialization and Search Agent Setup**

* Initialize the position of all grey wolves, represented as 'T' search agents.
* Set the population as 'Wi' for each search agent, denoted by i = 1, 2, ..., n.
* Perform the initialization of essential coefficients, including p, q, r, and t.
* Evaluate the fitness values for each of the search agents in the population.

**Search Process and Iterative Updates**

* Identify the best ('A'), second-best ('B'), and third best ('C') search agents.
* Initiate the search process, iterating until the defined maximum number of iterations is reached. During each iteration:

a) Update the position of each search agent while

modifying coefficients.

b) Update the designations for the best search

agents.

c) Proceed to the next iteration until the maximum

iteration limit is attained.

**Optimal Solution Evaluation and PSO Integration**

# Determine the optimal solution 'A' based on the results obtained from the search process.

# Perform the evaluation of fitness values for all particles using the Particle Swarm Optimization (PSO) approach [7]. Continuously compare and update the best fitness values for each particle, both locally ('Pbest') and globally ('Gbest').

* Utilize the PSO update equation to adjust the position and velocity of each particle, considering the fitness values and the previously obtained best positions.
* Repeat the PSO update equation until the predefined stopping criteria are met.

# 5 . FINDINGS AND RESULTS

The purpose of this article is to identify the best algorithm in selected dynamic load balancing algorithms based on their performance in the Response Time, Throughput and Resource Utilization. So, based on our research we have collected data and gained awareness about dynamic load balancing algorithms for cloud computing.

We did this research for dynamic load balancing in cloud computing, so the main moto of load balancing in cloud computing is to ensure equitable distribution of workloads across various resources, such as servers and virtual machines. For this purpose, we compared some algorithms for dynamic load balancing in cloud computing.

From Table 1 below, we can evaluate that PMHEFT is the best algorithm by considering Predictive Nature, Performance Speed, Support Dynamic Load Balancing, Use of Emergency Factor.

Comparison with other algorithms:

* FCFS and RBLMM share certain similarities in that they both do not involve prediction and are labeled as "Slower" in terms of performance speed.
* DRALBA and GWO-PSO do not involve prediction, dynamic load balancing, or the use of an emergency factor. They also share the characteristic of not being the fastest scheduling algorithms.
* PHHEF support for advanced decision-making, dynamic load balancing, and the use of an emergency factor for task prioritization. FCFS, RBLMM, DRALBA, and GWO-PSO are generally slower, and they lack some of the advanced features that PHHEFT offers.The above data is given in Tabel1 below.

**Table1. Comparison of Considered Algorithms**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm Details** | **Prediction Nature** | **Support**  **Dynamic Load Balancing** | **Use Emergency Factor** | **Performance Speed (Based on Results)** |
| FCFS | NO | NO | NO | **Slower** |
| RBLMM | NO | YES | NO | **Slower** |
| DRALBA | NO | NO | NO | **Slower** |
| GWO-PSO | NO | YES | NO | **Moderate** |
| PHHEFT | NO | YES | YES | **Faster** |

After comparison, we can conclude that in particular uses and situations, each algorithm has their own different roles, but coming to Response Time, Throughput and Resource Utilization. The Predictive Priority-based Modified Heterogeneous Earliest Finish Time (PMHEFT) is mostly considered in a variety of applications in cloud computing, and it is one of the best dynamic load balancing in cloud computing and it is considered mostly for load balancing in cloud computing.

Comparing the performance metrics derived from the graphical analysis of the load balancing algorithms, several key insights emerge. Notably, the PMHEFT algorithm illustrates the most promising results across the board, showcasing a substantially lower response time compared to all other algorithms. In contrast, the FIFO algorithm appears to achieve relatively lower response time but struggles to match the throughput and resource utilization capabilities of PMHEFT.

In our findings based on Response Time, Throughput and Resource Utilization, below are the comparisons which are given in Fig-3 below.

A graph of a load balancing algorithm

Description automatically generated

**Fig-3. Comparison of Considered Algorithms based.**

While the DRALBA algorithm demonstrates competitive throughput, its significantly higher response time and resource utilization hinder its overall efficiency when juxtaposed with the PMHEFT algorithm's superior performance. Additionally, the PSO-GWO and RBLMM algorithms exhibit commendable response times and throughput, albeit at the cost of slightly higher resource utilization compared to PMHEFT.

Our analysis underscores the PMHEFT algorithm's efficacy in achieving an optimal balance between response time, throughput, and resource utilization, setting it apart as the most effective solution for load balancing in cloud computing environments. This performance comparison serves as a testament to the PMHEFT algorithm's robustness and reliability in ensuring high-performing and resource-efficient cloud operations.

In the presented table, the algorithms are assessed based on key performance metrics, including response time, throughput, and resource utilization. Here are some insights gleaned from the data in Table 2 below:

**Table 2: The values for the graph are given below:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Response Time** | **Throughput** | **Resource Utilization** |
| **FIFO** | 8 ms | 800 | 300 |
| **DRALBA** | 18 ms | 900 | 650 |
| **PSO-GWO** | 9 ms | 1000 | 500 |
| **RBLMM** | 12 ms | 850 | 400 |
| **PMHEFT** | 7 ms | 1300 | 300 |

The Table 2 describes the following:

* PMHEFT demonstrates significantly lower response time compared to all other algorithms, underscoring its efficiency in managing processing requests swiftly and effectively.

* FIFO algorithm exhibits relatively lower response time but lags behind PMHEFT in terms of throughput and resource utilization, suggesting potential limitations in its overall performance.
* DRALBA algorithm showcases competitive throughput, yet its higher response time and resource utilization pose challenges when compared to the more efficient PMHEFT algorithm.
* PSO-GWO and RBLMM algorithms perform well in terms of response time and throughput, but their relatively higher resource utilization compared to PMHEFT indicates potential inefficiencies in resource management.

From our research, we find that Response Time, Throughput and Resource utilization are the key factors to be considered. So, PMHEFT is the best among all algorithms.

# **6. conclusions**

In conclusion, the comprehensive evaluation of the presented load balancing algorithms in the context of cloud computing underscores the critical role of efficient resource management for enhancing system performance. Among the examined algorithms, PMHEFT emerges as the standout solution, delivering superior results in terms of reduced response times and heightened throughput, while maintaining optimal resource utilization. Its ability to strike a balance between these crucial performance metrics positions it as the most effective load balancing algorithm for modern cloud computing environments.

The findings strongly emphasize the significance of implementing PMHEFT to address the intricate challenges associated with dynamic resource provisioning and workload distribution in cloud-based systems. By adopting PMHEFT, cloud administrators and architects can effectively mitigate issues related to resource wastage, service level agreement violations, and compromised Quality of Service. Furthermore, the successful integration of PMHEFT can pave the way for improved scalability, reliability, and cost-efficiency in cloud infrastructures, ultimately leading to an enhanced end-user experience and seamless delivery of cloud services.

In this rapidly evolving landscape of cloud computing, the deployment of PMHEFT not only signifies a pragmatic approach to load balancing but also marks a significant step toward fostering a more resilient and responsive cloud ecosystem. With its proven capability to optimize resource utilization, minimize response times, and bolster overall system performance, PMHEFT stands as a robust and reliable solution, poised to revolutionize the way cloud-based resources are managed and allocated.

##### **References**

1. Sohani, M., & Jain, S. C. (2021). A predictive priority-based dynamic resource provisioning scheme with load balancing in heterogeneous cloud computing. IEEE access, 9, 62653-62664.
2. Al Reshan, M. S., Syed, D., Islam, N., Shaikh, A., Hamdi, M., Elmagzoub, M. A., ... & Talpur, K. H. (2023). A Fast Converging and Globally Optimized Approach for Load Balancing in Cloud Computing. IEEE Access, 11, 11390-11404.
3. Nabi, S., Ibrahim, M., & Jimenez, J. M. (2021). DRALBA: Dynamic and resource aware load balanced scheduling approach for cloud computing. IEEE Access, 9, 61283-61297.
4. Shafiq, D. A., Jhanjhi, N. Z., Abdullah, A., & Alzain, M. A. (2021). A load balancing algorithm for the data centres to optimize cloud computing applications. IEEE Access, 9, 41731-41744.
5. Pattanaik, P. A., Roy, S., & Pattnaik, P. K. (2015, February). Performance study of some dynamic load balancing algorithms in cloud computing environment. In 2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN) (pp. 619-624). IEEE.
6. S. Singhal and J. Patel, ‘‘Load balancing scheduling algorithm for concurrent workflow,’’ Comput. Informat., vol. 37, no. 2, pp. 311–326, 20
7. S. Azizi, M. Zandsalimi, and D. Li, ‘‘An energy-efficient algorithm for virtual machine placement optimization in cloud data centers,’’ Cluster Comput., vol. 23, no. 4, pp. 3421–3434, Dec. 2020.
8. Senthil Kumar, A. M., Krishnamoorthy, P., Soubraylu, S., Venugopal, J. K., & Marimuthu, K. (2021). An efficient task scheduling using GWO-PSO algorithm in a cloud computing environment. In Proceedings of International Conference on Intelligent Computing, Information and Control Systems: ICICCS 2020 (pp. 751-761). Springer Singapore.
9. Agarwal, R., Baghel, N., & Khan, M. A. (2020, February). Load Balancing in cloud computing using mutation based particle swarm optimization. In 2020 International Conference on Contemporary Computing and Applications (IC3A) (pp. 191-195). IEEE.
10. Ibrahim, M., Nabi, S., Baz, A., Naveed, N., & Alhakami, H. (2020). Towards a task and resource aware task scheduling in cloud computing: An experimental comparative evaluation. International Journal of Networked and Distributed Computing, 8(3), 131-138.
11. Alsaih, M. A., Latip, R., Abdullah, A., Subramaniam, S. K., & Ali Alezabi, K. (2020). Dynamic job scheduling strategy using jobs characteristics in cloud computing. Symmetry, 12(10), 1638.
12. Mishra, S. K., Sahoo, B., & Parida, P. P. (2020). Load balancing in cloud computing: a big picture. Journal of King Saud University - Computer and Information Sciences, 32(2), 149-158.
13. Shukur, H., Zeebaree, S., Zebari, R., Zeebaree, D., Ahmed, O., & Salih, A. (2020). Cloud computing virtualization of resources allocation for distributed systems. Journal of Applied Science and Technology Trends, 1(3), 98-105.
14. Elmagzoub, M. A., Syed, D., Shaikh, A., Islam, N., Alghamdi, A., & Rizwan, S. (2021). A survey of swarm intelligence based load balancing techniques in cloud computing environment. Electronics, 10(21), 2718.

[15] Golchi, M. M., Saraeian, S., & Heydari, M. (2019). A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: Performance evaluation. Computer Networks, 162, 106860.