LoadAwareDistributor: An Algorithmic Approach for Cloud Resource Allocation

Ameya Chavan [1], Keyur Gandhi[2], Hetal Chauhan[3], Ajinkya Kamthe[4], Dr.Chitrakant Banchhor [5]

*[1]Department of Computer Engineering
Vishwakarma Institute of Information Technology, Kondhwa, Pune, 411048, Maharashtra, India .*
*[2]Department of Computer Engineering
Vishwakarma Institute of Information Technology, Kondhwa, Pune, 411048, Maharashtra, India.*
*[3]Department of Computer Engineering
Vishwakarma Institute of Information Technology, Kondhwa, Pune, 411048, Maharashtra, India.*
*[4]Department of Computer Engineering
Vishwakarma Institute of Information Technology, Kondhwa, Pune, 411048, Maharashtra, India.*
*[5]Department of Computer Science Engineering (AI)
Vishwakarma Institute of Information Technology, Kondhwa, Pune, 411048, Maharashtra, India.*

---***---

Abstract - *This research delves into the complexities of load balancing algorithms within cloud computing, offering a thorough examination and proposing the LoadAwareDistributor algorithm as an innovative solution.. The hybrid LoadAwareDistributor prioritizes virtual machines with lower CPU utilization, introducing efficiency improvements quantified through metrics such as VM utilization, cloudlet completion time, and scalability. Comparative analysis reveals approximately 2.68% enhancement over conventional round-robin methods. The study advocates for future advancements, emphasizing adaptability, autonomous balancing, and hybrid methodologies. It concludes by emphasizing the critical considerations in multi-cloud settings, including data governance, cost-effectiveness, and mitigating vendor lock-in risks. The unique insights provided contribute significantly to the evolving landscape of load balancing in cloud computing, paving the way for further innovations and addressing the dynamic challenges associated with optimizing resource allocations and ensuring scalability across diverse cloud environments.*

Key Words:

Algorithm, VM(Virtual Machine), Cloudlet, Cloudsim, load-balancing

I. INTRODUCTION

In the ever-evolving realm of cloud computing, the necessity for scalable and efficient solutions is crucial, with load balancing acting as a pivotal factor. Navigating the intricate landscape of diverse workloads in data centers, the challenge lies in optimizing resource usage and ensuring system responsiveness. This research delves into the nuances of load-balancing algorithms, not merely to dissect their complexities but to pave the way for innovation. The unveiling of the LoadAwareDistributor algorithm marks a significant leap forward, directing the discourse toward solution-driven strategies. By prioritizing virtual machines based on lower CPU utilization, this algorithm becomes a beacon for enhancing the efficiency of load distribution. In the evolving panorama of cloud environments, the incorporation of machine learning and nature-inspired algorithms becomes imperative. The proposed algorithm, with its emphasis on addressing load-balancing complexities, presents a promising avenue for future advancements. This research lays a robust foundation for a comprehensive exploration, underscoring the pivotal role of innovative load-balancing techniques in achieving seamless resource allocation and ensuring the scalability of systems within cloud computing infrastructures.

II. LITERATURE REVIEW

[1] Muhammad Asim Shahid et al. present an extensive review of existing algorithms and introduce novel techniques. The paper offers insights into fault tolerance, dynamic load balancing, and hybrid optimization approaches. It significantly contributes by addressing gaps in current literature, emphasizing the growing importance of load balancing in cloud environments. The research offers valuable perspectives on optimizing system performance and resource allocation, aligning with and enhancing the broader field of load-balancing strategies in cloud computing.

[2] Bayan et al. discuss the importance of load balancing in cloud computing and propose a priority-based load balancing algorithm to improve the performance of cloud computing applications. The

proposed algorithm aims to enhance load balancing by prioritizing tasks based on their deadline and assigning them to virtual machines accordingly. The algorithm is implemented in several stages, including reading user request data, assigning priority values to tasks, and scheduling tasks based on their priorities and resource utilization. The document also provides a comparative study of existing load balancing algorithms, such as round robin, equally spread current execution, active monitoring, and throttled load balancing. The proposed algorithm is expected to improve load-balancing performance and resource utilization in cloud computing applications.

[3] Kalpana et al. delve into optimizing cloud computing through load balancing, introducing the Enhanced Genetic Algorithm. Unlike existing methods, it minimizes execution time without additional hardware or complex steps. Results show superiority over the Improved Genetic Algorithm in response time, finish time, energy consumption, cost, and migrations. The algorithm's simplicity and efficiency, avoiding extra resources, distinguish it. Future work involves hybrid approaches and meta-heuristic algorithms. Compared to similar studies, this research emphasizes resource efficiency and fault minimization in dynamic cloud networks, offering a promising step toward enhanced load balancing and overall system optimization.

[4] K. Samunnisa et al. delve into the crucial domain of load balancing in cloud computing, emphasizing its significance in optimizing application performance and reliability. The literature review surveys various load balancing algorithms, such as Weighted Round Robin and Software-Defined Networking Adaptive. It categorizes these algorithms based on system load and system topology, detailing centralized, distributed, static, dynamic, and adaptive approaches. The study recognizes challenges in cloud load balancing, including scalability and efficient resource use. While providing an extensive overview, the literature lacks a unified analysis of multiple load-balancing techniques. The paper aims to bridge this gap by presenting a consolidated analysis of various algorithms, contributing to a better understanding of load-balancing complexities in cloud computing and offering insights for future research directions. The methodology involves comparing algorithms and classifying them based on system load and topology, ensuring a comprehensive examination of the existing landscape.

[5] Shahid et al. reveal a growing emphasis on optimizing overall response time, Data Center Processing Time (DCPT), and cost parameters. Noteworthy gaps include a need for innovative service broker policies integrating machine learning and artificial intelligence. The paper by [5] Shahid et al. significantly contributes by systematically evaluating PSO, RR, ESCE, and throttled algorithms across various service broker policies. Utilizing Cloud Analyst and proposing future integration with advanced technologies, the study stands out for its potential impact on evolving cloud computing strategies and addressing the growing complexities in large-scale applications.

[6] Foram F Kherani et al. investigate the crucial topic of load balancing in cloud computing, acknowledging the escalating challenge for cloud providers as user numbers increase. The literature review contextualizes the study by highlighting the significance of load balancing in enhancing cloud system performance and maintaining efficiency. Prior research, as exemplified by works such as, emphasizes load-balancing goals including performance improvement, system stability, flexibility enhancement, and fault tolerance. The classification of load balancing algorithms into static and dynamic types is explored, with the review showcasing diverse techniques such as Round Robin, Central Manager, Threshold, and randomized algorithms. Notable contributions from this introduce specific algorithms like Round Robin, Equally Spread Current Execution Load, and Throttled Load Balancing. Existing studies reveal the need for dynamic workload distribution and energy-efficient practices, aligning with the broader trends toward green computing. However, the literature review indicates gaps related to a comprehensive comparison of existing load-balancing algorithms in cloud computing.

[7] Jing He et al. explore load balancing in cloud computing, emphasizing the challenge of resource availability and deadlock prevention. It discusses the significance of load balancing in enhancing system performance and reducing job rejections. The paper compares static and dynamic load balancing, highlighting goals such as performance improvement and system stability. Three load balancing algorithms, including Round Robin, are discussed. The findings stress the importance of load balancing for achieving green computing and cite various algorithms in the literature. Overall, the paper contributes insights into load-balancing strategies, offering implications for cloud optimization.

III. COMMON LOAD BALANCING APPROACHES

A. First Come First Serve Algorithm:

When it comes to load balancing, the first come first serve (FCFS) algorithm is a straightforward scheduling technique. Because FCFS is not a preemptive scheduling method, a process runs continuously until it's completed after it's been assigned to the resource. For this to work, incoming tasks need to be queued up, and the first job is assigned to the available resource. This straightforward approach ensures that jobs are done one after the next, in the sequence in which they're received. However, FCFS may not always optimize load distribution, especially in dynamic situations where tasks run at different times.

Usually, more complex algorithms with better load balancing capabilities are used, such as round robin, weighted round robin, or shortest job first.

B. Shortest Job First Algorithm

The algorithm ranks jobs according to how long they should take to complete. In this scenario, SJF chooses the task with the least estimated processing time for instant execution to reduce the average waiting time for tasks. Reliability in task execution time estimation is critical to SJF's efficacy. If longer tasks are overstated, inaccurate task duration estimates may hurt the effectiveness of task distribution and prolong wait times for shorter activities. There are various drawbacks when using the Shortest Job First (SJF) algorithm for load

C. Priority-Based Algorithm

A priority-based algorithm in load balancing is a method that organizes tasks or processes based on assigned priorities to ensure efficient resource utilization and timely execution. This approach involves categorizing tasks according to their relative importance or criticality, allowing higher-priority tasks to be executed ahead of others. Priorities are typically assigned based on specific attributes, such as task urgency, importance, or deadline constraints. The basic principle of this algorithm lies in its ability to optimize the execution of tasks by prioritizing important tasks. By assigning priorities, the algorithm ensures that important and time-critical tasks arecompleted quickly, reducing overall latency and improving system responsiveness. Although this approach is effective when addressing critical task requirements, it can face challenges in scenarios where there is a list of many high- priority tasks, delaying the completion of lower-priority tasks. As a result, certain tasks or processes may run out of resources. Therefore, though priority-based algorithms improve the responsiveness and importance of critical tasks, a balance between task prioritization and fair resource allocation to ensure optimal load distribution is needed.

D.Round Robin Algorithm

The round-robin method delivers tasks to resources in a circular order, assuming uniform capabilities among resources and disregarding current load or availability. Each CPU processes a job in turn, with all processors sharing an equal burden. When the number of processors is much fewer than the number of processes, this strategy eliminates the requirement for inter-process communication. While it avoids the issue of priority, the defined time slots may result in certain processors being fully utilized while others are underutilized. Although an Adaptive Load Balancing technique solves workload changes, it may increase waiting time.

IV. SOME ADVANCED LOAD-BALANCING APPROACHES

A. Classification ML of tasks and VMs

In cloud computing, the classification of tasks and virtual machines (VMs) through machine learning is pivotal for enhancing system efficiency. Algorithms like k-means, hierarchical clustering, regression, and neural networks categorize tasks based on workload, enabling adaptive resource allocation across VMs.

This dynamic approach, informed by historical and real-time data, optimizes load balancing and improves overall system performance. Machine learning's role in addressing load unbalancing challenges is crucial, mitigating risks of VM overload or underutilization. Reinforcement learning and ensemble methods contribute to optimal load-balancing policies and task classification accuracy. The future holds promise for hybrid models, integration with emerging technologies like edge computing, and addressing scalability and real-time decision-making challenges, solidifying machine learning's central role in ongoing cloud computing system optimization.

B. Dynamic Load Balancing Algorithm

Dynamic load balancing plays a pivotal role in optimizing system efficiency through resource allocation.Ensuring that resources are used as efficiently as possible throughout the system and preventing certain nodes or servers from overloading, improves system performance as a whole. Furthermore, load balancing mechanisms ensure system stability despite possible failures or disruptions by distributing work among multiple nodes, thereby contributing to fault tolerance and resilience. Although dynamic load balancing has many advantages, it also has drawbacks. As workloads are distributed, it can be difficult to maintain consistency and coherence throughout the system, which raises concerns about synchronization between nodes. Developing algorithms that can effectively adjust to changing conditions is complex and makes it challenging to manage and modify load distribution in response to changing system requirements.One of the greatest advantages of dynamic load balancing is that if one node fails, it does not cause the failure of the system, but it has a ripple effect in that it influences the way the system performs.

C. Virtual Machine APC

The Virtual Machine Adaptive Predictive Control (APC) algorithm presents an advanced approach to load balancing in cloud computing. This algorithm, driven by adaptability and prediction, dynamically adjusts resource allocations based on real-time and predictive analyses of Virtual Machine (VM) performance metrics like CPU utilization and memory usage. Its proactive identification of potential imbalances ensures optimal resource distribution, effectively preventing bottlenecks and underutilization.

Despite challenges in accurately predicting workloads during sudden spikes, the Virtual Machine APC algorithm excels in maintaining consistent Quality of Service (QoS) for end-users, showcasing its effectiveness in handling the dynamic nature of cloud workloads. Positioned as a promising solution in load balancing strategies, it aligns with the overarching goals of equitable resource distribution, minimized latency, and optimized virtualized infrastructure utilization, contributing to efficient and responsive cloud computing environments.

D. Priority-Based Supervised Classification ML
Priority-Based Supervised Classification is an approach in machine learning where the classification model assigns priorities to different classes or labels. The objective is to not only predict the class of an input but also to prioritize the importance of each predicted class. This can be particularly useful in scenarios where certain classes have higher significance or impact than others. Priority-Based Supervised Classification in the context of load balancing involves using machine learning models to intelligently distribute incoming tasks or requests among available resources based on their priority levels. This approach optimizes resource utilization by considering the importance or significance of different tasks, aiming to achieve efficient load balancing in a system.

Priority-based supervised Classification allows load balancers to allocate resources more effectively by considering the priority levels of incoming tasks. This ensures that critical or high-priority tasks receive preferential treatment, optimizing resource utilization. The prioritization of tasks contributes to improved overall system efficiency. By focusing on high-priority tasks, the load-balancing system can streamline resource allocation, reducing processing delays and enhancing the system's responsiveness.

Priority-Based Supervised Classification in load balancing is a powerful approach that brings intelligence and adaptability to resource allocation decisions. When implemented effectively, it can contribute significantly to the efficient operation of systems, particularly in dynamic and demanding environments.

E. Using Queues Prioritize Task Scheduling

Using Queues to Prioritize Task Scheduling in load balancing involves organizing tasks into queues and implementing a scheduling mechanism that considers the priority of tasks in these queues. This approach ensures that tasks with higher significance or urgency are processed before others, contributing to efficient load balancing.

Prioritizing tasks through queues allows for optimized resource allocation. High-priority tasks are processed promptly, ensuring that critical operations receive the necessary resources and reducing resource contention. High-priority tasks are scheduled and processed quickly, leading to improved system responsiveness. This is particularly important in applications where timely processing is crucial for meeting user expectations.

Future developments may involve the creation of more sophisticated and adaptive scheduling algorithms. Machine learning techniques, including reinforcement learning and predictive analytics, could be applied to enhance the intelligence of task scheduling based on historical data and real-time conditions.As edge computing continues to gain prominence, the use of queues for task scheduling becomes crucial in decentralized and distributed environments. Prioritizing tasks becomes even more vital in edge scenarios where resources are constrained, and timely processing is essential.

F. Honey Bee-Based Load Balancing
The Honey Bee Algorithm (HBA) is inspired by honeybees' cooperative foraging habits to dynamically identify and share food sources to improve decision-making. When applied to cloud computing, it mimics adaptive and economical harvesting processes and improves resource allocation. Foraging bees actively exchange information. This is similar to maximizing the use of virtual machines in a cloud environment to minimize latency when server demand is dynamic. Honey bee behavior can be mapped as below:

TABLE I
DESCRIPTION OF HONEY-BEE HIVE TASKS IN CLOUD ENVIRONMENT

Honey-Bee Hive	Cloud Environment
Honey-Bee Hive	Represents a task in the cloud
Food Source	Virtual Machine
Searching (Foraging) for Food Source	The task is loaded to the VM
Honey-Bee getting crowded near the food source	Overloading state of VM
A New Food Source is Found	The task is removed from an overloaded VM and scheduled to another, underloaded VM

In 2018, [8] Ehsanimoghadam and Effatparvar improved the Honey Bee (HB) algorithm by introducing a job priority parameter to minimize the search time for work assignments. This change includes calculating virtual machine utilization and classifying public clouds into overloaded, balanced, and overloaded sectors. Virtual machine priority values are set based on cost and performance, but challenges remain in addressing similar priorities. Furthermore, [9] Kiritbhai and Shah addressed the prioritization dilemma by combining round-robin (RR) techniques with a hybrid strategy, thereby improving the quality of service, energy optimization, task scheduling, and Significantly improved load balancing. Further improvements to the Honey Bee algorithm include the integration of parallel versions, local search techniques, parameter tuning for convergence, and hybridization with other algorithms to increase flexibility and efficiency in various optimization scenarios.

V. METRICS OF LOAD BALANCING

A. Throughput

It is used to determine the number of assignments that have been completed, with higher throughput leading to better performance.

B. Fault Tolerance

A system can recover from failures or errors. It is crucial for the load balancing mechanism to demonstrate strong fault-tolerant capabilities.

C. Migration Time

Refers to the duration required for the relocation of tasks or resources between distinct nodes within a network.

D. Response Time

It is the measure of time that is taken by a specific load-adjusting calculation to react to an assignment in a framework. This parameter ought to be limited for better execution of a framework.

E. Scalability

A calculation can perform Load adjusting for any limited number of hubs of a framework. This metric ought to be enhanced for a decent framework.

F. CPU Usage

It is the percentage of CPU processing power that is actively used. Effective load balancing aims to distribute CPU load evenly across available resources.

G. Memory Usage

It is the percentage of available memory used by the system. It is used to evaluate how well the algorithm manages memory-intensive tasks, prevents memory bottlenecks, and ensures optimal resource utilization.

H. Algorithm Overhead

It measures the additional computing resources consumed by the load-balancing algorithm itself. This should be minimal, ensuring efficient utilization of resources and optimal system performance.

VI. BASE ALGORITHMS USED

A. Load-Aware load-balancing

Load-Aware Load Balancing is a dynamic and sophisticated approach used in distributed computing environments to optimize resource utilization and maintain system efficiency. Because load balancing considers more than simply CPU, memory, network traffic, and application-specific parameters, it differs from traditional load balancing. Rather, it assesses the overall health of the system and distributes workloads or tasks automatically according to available resources. Enhancing system performance, preventing bottlenecks, and guaranteeing optimal resource utilization are the objectives of this sophisticated method. Load-Aware Real-time indicators are continuously monitored and analyzed via load balancing. By dynamically assigning tasks to resources that are less busy or more appropriate for the work at hand, it adapts to changing circumstances.

This flexibility allows the system to adjust to changes in demand, scale as needed, and maintain efficient performance even in peak loads. The emphasis remains on using insights into the state of resources to optimize performance and improve the overall resiliency and scalability of distributed systems.

B. Round Robin Algorithm

Refer III.D above

C. Least Loaded VM

Least Loaded VM Load Balancing is a technique used in cloud computing to distribute tasks or workloads across virtual machines (VMs) in a way that ensures each VM has a similar and minimal load compared to others. The main objective is to balance workloads among all virtual machines (VMs) in the cloud to prevent any one VM from using resources excessively or insufficiently. This method entails keeping an eye on and evaluating the workload and resource usage of every virtual machine. To keep the overall balance of the system, tasks or jobs are given to the virtual machine (VM) with the least workload. This strategy helps prevent performance bottlenecks or overcrowded virtual machines (VMs) in cloud architectures, minimizes response times, and optimizes resource consumption by continuously evaluating each VM's workload and dynamic job assignment based on the machine's capacity.

VII. PROPOSED LOAD BALANCING ALGORITHM

We have proposed a hybrid load balancing algorithm whose objective is to distribute the load more evenly and prioritize VMs with lower CPU utilization. It is as follows.

A. Resource Characteristics Processing

1)The algorithm begins by processing the resource characteristics of data centers. When characteristics from all data centers are received, it triggers the VM distribution process.

B. VM Distribution using Round Robin

1) The algorithm utilizes a round-robin approach to distribute VMs among available data centers.

2) For each VM, it selects a data center in a round-robin manner from the list of available data centers.

3) The algorithm attempts to create the VM in the chosen data center by sending a VM_CREATE_ACK event.

C. Cloudlet Submission and Load Balancing

1) The algorithm overrides the submitCloudletList method inherited from the DatacenterBroker class.

2) For each cloudlet in the submitted list:

a. It identifies the least loaded virtual machine by calling a method named getLeastLoadedVm.

b. It binds the current cloudlet to the least loaded virtual machine.

D. Finding the Least Loaded VM

1) The algorithm includes a method named getLeastLoadedVm responsible for finding the least loaded virtual machine.

2) It iterates through the list of virtual machines and initializes the least loaded VM as the first VM in the list.

3) For each VM:

a. It calculates the expected completion time for the current VM using a method named getExpectedCompletionTime.

b. It compares the expected completion time of the current VM with the minimum expected time found so far.

c. If the current VM has a lower expected completion time, it updates the least loaded VM.

4) The method returns the VM with the minimum expected completion time.

E. Expected Completion Time Calculation

1) The algorithm assumes the existence of a method named getExpectedCompletionTime to calculate the expected completion time for a given virtual machine.

2) The algorithm calculates the expected completion time for a given virtual machine (VM) by summing the lengths of all associated cloudlets and dividing it by the VM's MIPS (Million Instructions Per Second) capacity.

VIII. SIMULATION TOOL AND EXPERIMENTAL RESULTS

For our experimentation, we employed the CloudSim framework, a robust and widely-used simulation tool designed specifically for modeling and simulating cloud computing environments. CloudSim provides a flexible and extensible platform that allows researchers to simulate diverse cloud scenarios, making it an ideal choice for our study.

IX. RESULTS AND DISCUSSION

In our investigation, we evaluated the proposed load balancing algorithm, within the CloudSim simulation environment. The key metrics assessed include:

1) VM Utilization : We measured the CPU utilization of VMs to gauge how effectively the load balancing algorithm distributes computational load across the available virtual machines.

2) Cloudlet Completion Time : The time taken for each cloudlet to complete its execution was recorded. This metric reflects the efficiency of task execution and overall system performance.

3) Scalability : To assess its scalability, the algorithm's performance was evaluated under varying workloads and system sizes. This involved increasing the number of cloudlets and VMs to observe how well the algorithm adapts to changing conditions.

4) Comparison with Baseline : We compared the results of our algorithm with a baseline approach, such as traditional round-robin load balancing, to highlight the improvements achieved.

The experimental setup involved creating a realistic cloud environment with multiple data centers, virtual machines, and cloudlets. Through rigorous experimentation and analysis of the aforementioned metrics, we aimed to provide insights into the effectiveness and efficiency of the proposed algorithm in load balancing.

TABLE II

VM SPECIFICATIONS

VM ID	VM MIPS	VM Image Size	Memory (GB)	No. of CPU	VMM
0	1000	1000	1024	1	Xen
1	1050	1000	2048	1	Xen
2	1100	1000	3072	1	Xen
3	1150	1000	4096	1	Xen
4	1200	1000	5120	1	Xen
.	.	.	.	.	.
14	1700	1000	15360	1	Xen

TABLE III

TASK SPECIFICATIONS

Task ID	Length	File Size	Output File	No. of CPU
0	10000	300	300	1
1	10020	300	300	1
2	10040	300	300	1
3	10060	300	300	1
4	10080	300	300	1
.	.	.	.	.
29999	10980	300	300	1

TABLE IV

ALGORITHM PERFORMANCE COMPARISON

Algorithm	Total CPU Time (s)	Average CPU Time (s)
First Come First Serve	7073276.25668	235.77588
Priority Based	7073304.15577	235.77681
Shortest Job First	7073432.98846	235.78110
Round Robin	7073432.98846	235.78110
Proposed Algorithm	6888657.00668	229.62190

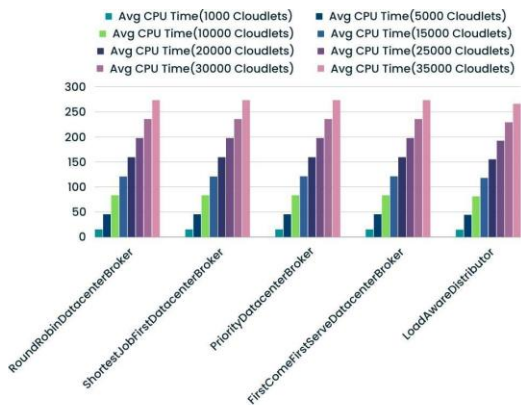

Fig. 1. Average CPU Time Comparison for different no. of cloudlets with different Algorithms.

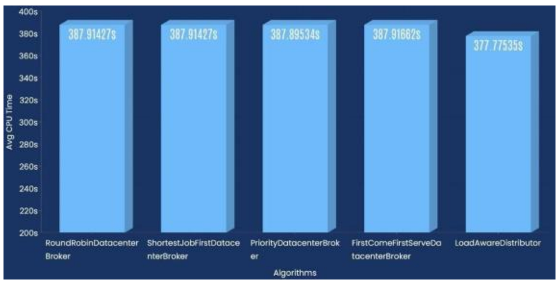

Fig. 2. Average CPU Time Comparison for a single cloudlet(i.e 50000) with different Algorithms.

We found out that the proposed algorithm is 2.63% more efficient than the round-robin algorithm.

XI. FUTURE WORK

The future improvements for the proposed algorithm and its load balancing in cloud computing entail the fusion of machine learning for adaptability, delving into autonomous and self-learning balancing mechanisms, and pioneering hybrid approaches. When implementing load balancing in multiple or hybrid cloud settings, there are a few crucial factors to consider. Even though the data is less sensitive, it is still crucial to strictly adhere to data governance and regulatory requirements. This entails thinking about data residency, privacy regulations, and meeting regulatory requirements. Moreover, it is essential to consider the costs associated with data transfer between clouds to guarantee cost-effectiveness. Moreover, there is a chance of vendor lock-in hazards even with less sensitive data. Heavily relying on one or more cloud services could limit future migration flexibility. Finally, administering several public clouds successfully demands a wide range of skills within the ecosystem of each provider. This means handling procedures, monitoring output, and maintaining security guidelines in diverse cloud settings. More explorations include enhancing energy efficiency, integrating with edge and fog computing, and connecting with emerging technologies such as quantum computing and blockchain. Dynamic workload migration, real-time analytics, and security-aware strategies are focal points for improving system performance and security. Standardizing metrics, integrating cloud-native technologies, and addressing cross-domain scenarios are critical elements for continuous innovation, optimizing resource allocations, and ensuring scalability across diverse cloud environments.

XI. CONCLUSION

This paper focuses on the popular load-balancing algorithms in today's cloud environment, analyzing and proposing an improved algorithm (LoadAwareDistributor) based on an algorithm already in place to improve load balancing over Round Robin, and has accomplished the following goals. It provides an extensive review and guide for load balancing in cloud computing, to address gaps in understanding the complexities associated with load balancing. It delves into the factors contributing to load unbalancing, offering an overview of various approaches, classifying techniques, and examining the challenges encountered by researchers. Moreover, it underscores the significance of machine learning, dynamic load balancing, and nature-inspired algorithms in tackling load-balancing complexities. It highlights the importance of efficient resource utilization, system stability, and responsiveness in cloud environments.

The proposed algorithm aims to improve load balancing in cloud computing by evenly distributing the workload and prioritizing virtual machines with lower CPU utilization. Comparative analysis reveals approximately 2.68% enhancement over conventional round-robin methods. Using a hybrid approach involving resource analysis, round-robin virtual machine allocation, and cloudlet submission for load balancing, the algorithm is evaluated based on metrics like VM utilization, cloudlet completion time, and scalability. Comparative analysis with a baseline approach highlights improvements. It shows potential for enhancing load balancing performance and resource utilization in cloud applications. Further research is needed to fully assess its effectiveness.

REFERENCES

[1] Muhammad Asim Shahid, Noman Islam, Muhammad Mansoor Alam, Mazliham Mohd Su'ud, And Shahrulniza Musa, "A Comprehensive Study of Load Balancing Approaches in the Cloud Computing Environment and a Novel Fault Tolerance Approach", IEEE Access: The Multidisciplinary Open Access Journal, Vol. 8,2020, Digital Object Identifier 10.1109/ACCESS.2020.3009184.

[2] Bayan A. Al Amal Murayki Alruwaili, Manoona Humayun, NZ Jhanjhi, "Proposing a Load Balancing Algorithm for Cloud Computing Applications", International Conference on Recent Trends in Computing,2021,doi:10.1088/1742-6596/1979/1/0 12034.

[3] Kalpana, Manjula Shanbhog, "Load Balancing in Cloud Computing with Enhanced Genetic Algorithm", International Journal of Recent Technology and Engineering (IJRTE), Vol. 8, July 2019, DOI: 10.35940/ijrte.B1176.0782S619.

[4] K. Samunnisa, G. Sunil Vijaya Kumar, K. Madhavi, "A Circumscribed Research of Load Balancing Techniques in Cloud Computing", Inter- national Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol. 8,2019,DOI: 10.35940/ijitee.F1068.0486S419.

[5] Shahid, M.A.; Alam, M.M.; Su'ud, M.M. Performance Evaluation of Load-Balancing Algorithms with Different Service Broker Policies for Cloud Computing. Appl. Sci. 2023, 13, 1586. https://doi.org/10.3390/ app13031586

[6] Foram F Kherani, Prof.Jignesh Vania, "Load Balancing in cloud computing", IJEDR, Vol. 2,2014.

[7] Jing He, "Cloud Computing Load Balancing Mechanism Taking into Account Load Balancing Ant Colony Optimization Algorithm", Hindawi Computational Intelligence and Neuroscience,2022, https://doi.org/10. 1155/2022/3120883

[8] Ehsanimoghadam, P., Effatparvar, M., 2018. Load balancing based on bee colony algorithm with partitioning of public clouds. Int. J. Adv. Comput. Sci. Appl. 9 (4) 450–455. https://doi.org/10.14569/IJACSA. 2018.090462

[9] Kiritbhai, P.B., Shah, N.Y., 2017. Optimizing Load Balancing Technique for Efficient Load Balancing. Int. J. Innov. Res. Technol. 4 (6), 39–44.