Decision Making and Artificial Intelligence Trends

Volume 1, Issue 1, 77-90.

Optimization of load balancing in cloud computing using the bin-stretching approach

https://doi.org/10.22034/dmait.2024.203335

Amirreza Zare[1]*, Hossein Eetedadi[2]

1. Computer Engineering Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, IRAN. *Corresponding Author: amrrz.z@aut.ac.ir
2. Computer Engineering Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, IRAN.

ARTICLE INFO

Article history:
Received: 2024/07/20
Revised: 2024/08/14
Accepted: 2024/08/23
Published: 2024/09/30

Keywords:
Task, Resource utilization, Scheduling, Worst-fit, Cloud computing

ABSTRACT

Cloud computing has revolutionized how organizations and individuals utilize computational resources by offering scalable, flexible, and cost-effective solutions. However, one of the most pressing challenges within this domain is load balancing, which entails the optimal distribution of user tasks across virtual machines (VMs) to enhance efficiency and reduce latency. Traditional load-balancing algorithms, such as Min-Min and Max-Min, primarily focus on static allocation methods and are limited by their inability to adapt to dynamic workloads and varying resource availability. This paper introduces a novel Worst-Fit Based Load Balancing Algorithm (WFBLBA) to address these limitations by integrating memory management into the load-balancing process. WFBLBA models load balancing as a bucket-pulling problem and incorporates virtual memory considerations using a worst-fit heuristic approach. This innovative algorithm aims to minimize the makespan and improve resource utilization by considering the available RAM of VMs and the impact of virtual memory delays on task processing. The research methodology includes comprehensive simulations conducted using the CloudSim Plus framework, which facilitates a detailed analysis of the proposed algorithm's performance under various cloud computing scenarios. Our results indicate that WFBLBA significantly enhances resource efficiency by up to 20% and reduces task processing delays by 15% compared to existing algorithms. This improvement is attributed to WFBLBA's ability to optimally allocate tasks based on real-time resource availability and the dynamic requirements of cloud-based services. Furthermore, this paper discusses the implications of our findings for cloud service providers and users. By addressing the critical gap of RAM consideration in load balancing, WFBLBA offers a robust solution that can adapt to the fluctuating demands of cloud environments, thereby improving

overall service quality and user satisfaction. Future work will focus on refining the algorithm to include weighted bucket sizes based on task priority and VM capabilities, as well as expanding the simulation framework to test the algorithm's applicability in diverse cloud configurations. In conclusion, the WFBLBA presents a significant advancement in the field of cloud computing, providing an effective means of optimizing task scheduling and resource allocation. This contribution not only addresses the current shortcomings in load balancing strategies but also paves the way for more efficient and reliable cloud computing infrastructures.

1. Introduction

Cloud computing is a TCP/IP-based computing architecture (et al., 2021) that integrates various computational resources such as microprocessors, heavy processing units, robust memories, and similar elements (Khan, 2024). Today, cloud computing is recognized as a distributed computing environment (Al-Shareeda et al., 2024), providing resources over the internet to users and organizations (Neetu Settia et al., 2024), with costs calculated based on the consumption of these resources, allowing users to utilize these services as per their needs (Shafiq et al., 2022). In most applications, cloud computing can reduce costs, as cloud service providers can allocate resources optimally, benefiting themselves while users avoid the need for expensive software and hardware (Pradhan & Bisoy, 2022). Virtualization plays a crucial role in cloud computing by reducing operational costs in data centers and subsequently increasing resource utilization efficiency (Gogineni & MS, 2024). By consolidating multiple virtual machines on a single server or host in data centers, virtualization reduces costs and energy consumption for cloud service providers. The interaction between human applications and the Internet of Things (IoT) through cloud services is critical due to the need to process and analyze large and complex data generated by IoT devices.

Cloud services enable real-time storage and processing of vast amounts of data, allowing IoT devices to communicate seamlessly and effectively with each other and with human users. These interactions can enhance efficiency and productivity in many areas, such as healthcare, smart transportation, smart homes, and various industries. For example, in healthcare systems, IoT devices can continuously collect patient data and send it to the cloud for analysis and treatment recommendations, improving service quality and reducing costs (Abdulkareem et al., 2021). As mentioned, due to increasing demands (Shafiq et al., 2021) and the consequent growth of cloud-based service users, the scheduling of user requests is a critical and significant issue(Mohanty et al., 2021)(Meera & Valarmathi, 2024). Scheduling impacts users' needs, such as latency, and significantly affects cloud service providers by increasing resource utilization efficiency [20]. This necessitates the concept of dual scheduling [21]. A cloud computing scheduling approach that considers appropriate task scheduling and load balancing. Optimal task allocation to suitable virtual machines and optimal load balancing aim to meet the needs of both users and cloud service providers. As stated, time is a crucial requirement for users of cloud

services (Dhabliya et al., 2024). Therefore, the return time parameter is introduced as a key factor (Jena et al., 2022).

Return time, the interval between the start of the first task and the end of the last task (Lhomme et al., 2024), is a critical indicator of throughput, representing the number of tasks completed in a given period (Devaraj et al., 2020). Addressing both user and provider needs is essential [26], particularly in enhancing resource utilization efficiency (Thakur & Goraya, 2022). Cloud service providers aggregate resources and use virtual machines to serve users (Haris & Zubair, 2022). Proper task allocation and request handling, coupled with load balancing among virtual machines, can increase efficiency. Overall, the goal of designing related algorithms is to meet user and cloud service provider needs, reduce return time, and increase resource efficiency (Alsubaei et al., 2024). Existing algorithms fall into two main categories: static and dynamic (Khare et al., 2022). Static algorithms create a fixed schedule before virtual machines begin, aiming to reduce return time by minimizing task shifting between virtual machines. In contrast, dynamic algorithms move tasks between virtual machines during execution, aiming to maximize resource utilization despite the overhead from task shifting. Load balancing is an NP-Hard problem (Geetha et al., 2024), with no efficient polynomial algorithm yet available. The article (Dhahbi et al., 2021) presents a load-balancing algorithm addressing both user and provider needs simultaneously. Our goal in this paper is to improve the algorithm proposed by (Dhahbi et al., 2021) by modifying its structure and reducing return time. RAM is considered a key parameter, and the proposed algorithm modifications achieve the desired improvements.

The innovation of this paper lies in the integration of RAM considerations into the load-balancing process, a factor often overlooked in existing methodologies. Our proposed WFBLBA addresses this critical gap by incorporating virtual memory delays and RAM availability into the decision-making process for task allocation. By modeling load balancing as a bucket-pulling problem and applying the worst-fit heuristic, WFBLBA allocates tasks to VMs not only based on CPU capacity but also considering the available RAM, thus reducing the reliance on virtual memory and minimizing associated delays. This novel approach provides several key advantages over existing algorithms:

- Improved Resource Utilization: By considering both CPU and RAM capacities, WFBLBA ensures that tasks are allocated to the most suitable VMs, thereby increasing the overall resource utilization efficiency by 20%, as evidenced by our simulation results.
- Reduced Latency: The algorithm significantly reduces task processing delays by 15% by minimizing the use of virtual memory, which is a common bottleneck in traditional load-balancing methods.
- Scalability and Adaptability: WFBLBA is designed to adapt to the fluctuating demands of cloud environments, making it scalable across different cloud configurations and capable of handling varying workload patterns.

In conclusion, the WFBLBA represents a significant advancement in the field of cloud computing, providing a robust and innovative solution to the challenges of load balancing. By filling a critical gap in the literature and demonstrating clear quantitative improvements, this paper contributes to the ongoing efforts to enhance the performance and reliability of cloud services. The rest of the paper is structured

as follows: Section 2 reviews related work on load balancing in cloud computing. Section 3 explains the load balancing problem in the cloud environment. Proposed solutions are discussed in Section 4. Section 5 examines implementation and simulation details. Conclusions and findings are presented in Section 6.

2. Related Works

In this section, we aim to review some past research on load balancing and scheduling issues in cloud computing. Several approaches have been proposed to address load balancing and task scheduling problems in cloud computing. In the field of cloud computing, load balancing remains a critical challenge, significantly impacting the efficiency and scalability of cloud services. Traditional approaches to load balancing have predominantly been divided into static and dynamic algorithms. Static algorithms, such as the Min-Min and Max-Min algorithms, predefine task schedules based on prior knowledge of task and resource characteristics. While these algorithms are straightforward and provide baseline efficiency, they often fall short in environments where workload variability and resource availability are dynamic. Dynamic load balancing algorithms emerged to address the limitations of static approaches by allowing task migration during execution. Examples of such algorithms include those by Kang et al., which optimize for task heterogeneity and use heuristic methods to improve resource allocation. These dynamic algorithms have demonstrated improvements in resource utilization and reduction in completion times. However, they are often accompanied by increased computational overhead and complexity, which can be prohibitive in real-time applications and large-scale cloud infrastructures.

Recent advancements in load balancing have explored the use of machine learning and heuristic methods to predict and classify workload patterns, aiming to optimize task distribution more effectively. For instance, deep learning techniques have been employed to classify incoming traffic, while clustering methods have been used to group similar tasks and VMs. These methods offer a higher degree of adaptability but often lack consideration for the underlying hardware constraints, particularly RAM availability and virtual memory delays, which are crucial in ensuring optimal task execution and resource utilization. In (S. K. Mishra et al., 2020), a comprehensive review of approaches used for load balancing is provided. The initial research to tackle these issues derived from heterogeneous computing systems and grid computing. For instance, the Min-Min and Max-Min algorithms are used for task scheduling and are typically considered as a benchmark. The main idea, as presented in (Maheswaran et al., 1999), is that a virtual machine is assigned to a task which, if not allocated, could result in the most significant damage. In (Dhahbi et al., 2021), two static load balancing algorithms are proposed: the WFBLBA algorithm, based on the worst-fit algorithm for the bin packing problem, and the WFDBLBA algorithm, based on a reduced version of the worst-fit algorithm. In these algorithms, load balancing is modeled as a bucket-pulling problem, a variation of the bin-packing problem (Lhomme et al., 2024).

Recent research by Kang et al. (Gogineni & MS, 2024) focuses on task heterogeneity. The authors aimed to optimize build time, financial costs, and resource usage. The algorithm comprises two phases: in the first phase, tasks are allocated to virtual machines so that the completion time does not exceed the optimal completion time, which is the minimum completion time across all virtual machines. In the

second phase, any unassigned task from the first phase is allocated to virtual machines that offer the earliest completion time (Dhahbi et al., 2021). Experimental results show that this heuristic algorithm outperforms the results from (Banerjee et al., 2015) (Djebbar & Belalem, 2016) (Adhikari & Amgoth, 2018). The study (Punitha & Mala, 2021) proposes a deep learning technique to classify incoming traffic for efficient load balancing in server clusters. Additionally, (Sharma & Bala, 2020) employs a modified k-means clustering technique to categorize tasks and virtual machines. Li et al. (M. Li et al., 2020) proposed a load-balancing method based on a distributed machine-learning model to address performance discrepancies among servers.

Moreover, (Elsakaan & Amroun, 2024) introduce a novel approach to improving load balancing and task scheduling in cloud computing. This paper proposes a multi-level hybrid algorithm consisting of various stages to optimize task and resource allocation in cloud data centers. The algorithm is divided into two main parts: in the first stage, the Analytical Hierarchy Process (AHP) is used to rank tasks based on length and execution time. Tasks are then placed in a queue based on this ranking. In the second stage, the BATS+BAR optimization method is used to allocate resources such as CPU, memory, and bandwidth to tasks. Unlike existing systems, this method classifies incoming tasks based on size and complexity and uses a preventive approach to manage excess load on virtual machines (VMs). If a VM is overloaded, the system transfers tasks to idle VMs. This method uses the divide-and-conquer technique to split tasks into smaller parts and allocate them to different VMs. The results indicate that the proposed algorithm (Elsakaan & Amroun, 2024) improves efficiency and reduces system response time. Despite the positive aspects, this paper also has significant criticisms. The implementation complexity of the algorithm, like other hybrid algorithms, requires intricate setup and precise tuning. Additionally, more extensive testing is needed to verify its effectiveness in different environments and cloud settings and generalize the findings.

The study (Saravanan et al., 2023) introduces a hybrid algorithm called IWHO, which uses Wild Horse Optimization and Lévy Flight techniques to improve task scheduling and reduce costs in cloud computing environments. This paper's strengths include innovation in combining two optimization algorithms, reducing the time and cost needed for task execution, and optimizing energy consumption. However, like (Elsakaan & Amroun, 2024), the complexity of implementing this algorithm can be problematic for some systems, and the experimental results have limitations in generalizability to larger and more complex environments. By leveraging advanced optimization methods, this algorithm offers more effective solutions for task scheduling and load balancing in cloud data centers, potentially enhancing the performance of cloud computing systems significantly.

Past research on load balancing and task scheduling in cloud computing has significantly contributed to improving the efficiency and productivity of these systems. From simple static algorithms to advanced machine learning models, each has its unique advantages and disadvantages. Recent studies focusing on task heterogeneity and machine learning techniques have achieved notable improvements in cloud computing performance. However, there is still a need for further research and development in this field to achieve more optimal and efficient methods. In the subsequent sections, we aim to present and define the specific issue discussed in this article and propose an appropriate solution.

3. Problem Definition

In this section, we will examine the system model considered in this article and precisely define the issue under discussion. First, we will review the components that make up the system model. As mentioned in previous sections, the system model comprises a set of tasks that will be sent to virtual machines for processing. The set $T = \{T_1, T_2, \dots, T_n\}$ consists of n independent tasks without specified completion times and without priority. Additionally, the set $V = \{vm_1, vm_2, \dots, vm_m\}$ comprises m virtual machines, each with its own processing characteristics. As previously mentioned, the objective of this article is to allocate tasks to virtual machines in such a way that the makespan is minimized and resource efficiency is maximized. Let's delve into more details about the system model.

3.1. *Basic Definitions and Notations*

After a general understanding of the system model considered in this article, the problem's notations can be seen in Table 3-1. It is worth mentioning that the processing speed refers to the speed of each core, measured in millions of instructions per second (MIPS). Additionally, RAM refers to the main memory, measured in megabytes (MB) (Dhahbi et al., 2021). As observed, the variables stated in Table 3-1 help achieve a trade-off between the two stated objectives. These variables play a significant role in both increasing resource utilization efficiency and reducing the makespan, which is highly impactful for users. Evidently, to reach an optimal point, a suitable trade-off must be considered between these evaluation parameters. Next, we will examine the evaluation parameters in more detail.

Table 1: Notations

Notations	Type	Definition
m	Parameters	Number of virtual machines
n	Parameters	Number of Tasks
PE_{vm_j}	Parameters	Number of cores in vm_j
PS_{vm_j}	Parameters	Computing speed in vm_j
RAM_{vm_j}	Parameters	The amount of Ram in vm_j
L_{t_i}	Parameters	Length of task instruction t_i (in million units)
S_{t_i}	Auxiliary Variables	Queuing time to process task t_i
E_{t_i}	Auxiliary Variables	Time to start processing task t_i
PE_{t_i}	Decision Variables	The number of PEs required by task t_i
RAM_{t_i}	Decision Variables	Amount of RAM requested by task t_i

In Table 1, "Parameters" are fixed inputs or constants in the model, representing the characteristics of the system and tasks. They are not subject to change within the problem's context and are used to define the environment or constraints. "Auxiliary Variables" are used to support the calculations and logic of the model. They may be derived from parameters and decision variables but do not directly impact the final decision-making process. "Decision Variables" are the variables that the algorithm aims to determine or optimize. They represent choices that can be made within the model to achieve the desired outcome (e.g., task allocation and resource utilization).

3.2. *Performance Metrics*

In this section, we will discuss in detail the evaluation parameters of the problem, including completion time, makespan, resource utilization, and waiting time.

- **Completion Time of Virtual Machine**

The completion time of each virtual machine refers to the total expected execution time of the tasks assigned to the virtual machine. To better understand this parameter, it is necessary to have a good grasp of the concept of expected execution time. If we consider the task t_i and virtual machine vm_j, the expected execution time is equal to the ratio of the task's instruction length to the execution speed of the virtual machine, as shown in equation 3-1 (Dhahbi et al., 2021).

$$EET_{ij} = \frac{L_{t_i}}{PS_{vm_j}} \tag{3-1}$$

It is worth mentioning that equation 3-1 applies to situations where the scheduling method is considered as spatial sharing. If we consider the scheduling approach as time-sharing, equation 3-2 can be applied (Dhahbi et al., 2021).

$$EET_{ij} = \frac{L_{t_i} \times PE_{t_i}}{PS_{vm_j} \times Min(PE_{t_i}, PE_{vm_j})} \tag{3-2}$$

It is obvious that if $PE_{t_i} \leq PE_{vm_j}$, then, equation 3-1 and equation 3-2 will be equivalent, and otherwise, equation 3-2 should be considered (Dhahbi et al., 2021). Now, the completion time can be expressed as equation 3-3.

$$CT_{vm_j} = \sum_{i=1}^{m} EET_{ij} \tag{3-3}$$

- **Makespan Metric**

As mentioned in previous sections, the maximum completion time of tasks on virtual machines is called makespan, which we denote by the symbol MS in this paper (Dhahbi et al., 2021).

- **Resource Utilization Metric**

As stated, besides user requirement parameters, resource utilization is an important metric for cloud service providers. Resource utilization can be calculated using equation 3-4, which represents the percentage ratio of the total completion time of virtual machines to m times the makespan. Resource utilization is denoted by RU, as shown in equation 3-4 (Dhahbi et al., 2021).

$$RU_{T,V} = \frac{\sum_{j=1}^{m} CT(vm_j)}{MS(T.V) * m} * 100 \quad (3\text{-}4)$$

- **Waiting Time Metric and Average Waiting Time**

The final parameter we will examine is the waiting time of tasks. The waiting time of a task is the time spent in the queue before starting execution on the vCPU. The average waiting time is equal to the total waiting time of all tasks divided by the total number of tasks, as shown in Equations 3-5 (Dhahbi et al., 2021).

$$WT_{t_i} = S_{t_i} - E_{t_i} \quad (3\text{-}5)$$

After becoming familiar with and introducing all aspects of the system model and the relationships considered in this article, we aim to precisely and transparently examine the proposed solution in this paper.

4. Algorithm and Requirements

As mentioned in previous sections, the main idea of this paper is to include the RAM. In this section, considering the definition of the problem and the system model described in previous sections, we aim to examine the proposed solution of this paper in detail. As mentioned earlier, the main idea of this paper is to incorporate RAM in the load-balancing algorithm based on the worst fit, WFBLBA. Figure 4-1 illustrates the system architecture where our algorithm is implemented. As observable, it consists of four main components. The first component manages a global queue of all incoming tasks. The second component distributes tasks to the local queues of virtual machines (VMs) based on our proposed algorithm. The third component manages VMs that start executing assigned tasks using WFBLBA/WFDBLBA scheduling. The fourth component relates to the physical machines running the VMs.

4.1. *Algorithm Functionality*

This section will examine the algorithm used in this paper in detail. The WFBLBA algorithm used in this paper is based on the bin stretching problem, which will be briefly explained for better understanding. In this algorithm, we assume there are n items with sizes in the range (0,1] and m empty bins with infinite capacity. The goal of this problem is to place the items in the bins in such a way that the minimum number of bins is used. Accordingly, in the worst-fit method, each incoming item will be placed in the bin with the largest remaining capacity, and as stated, the WFBLBA algorithm will also be based on this principle (Dhahbi et al., 2021).

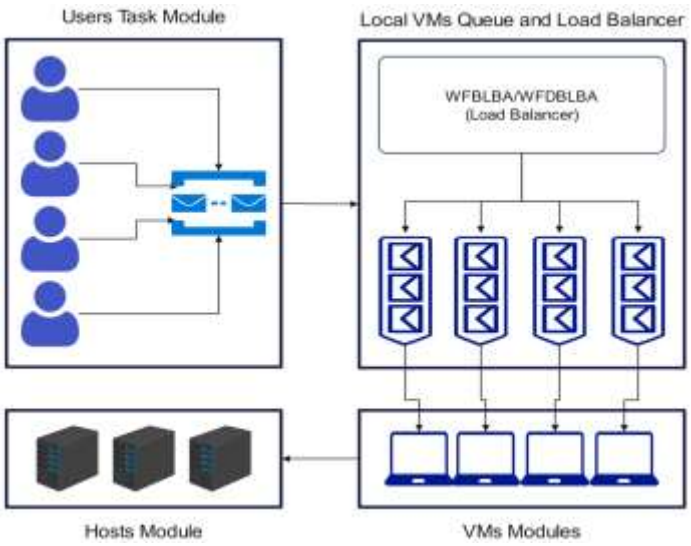

Figure 4-1: System Architecture

Now, we intend to precisely describe the steps of the stated algorithm and examine its details. This algorithm generally consists of five distinct steps.

Step One: Calculate the total length of all tasks, named TotalLength, and the total processing speed (PS) of all virtual machines in the variable TotalMIPS.

Step Two: For each virtual machine, such as vm_j, we consider a bin b_j with a capacity (size) according to equation 4-1.

$$size_{b_j} = TotalLength * \frac{PS_{vm_j}}{TotalMIPS} \quad (4\text{-}1)$$

Step Three: The i-th task is read from the task queue.

Step Four: The i-th task is assigned to the machine with the bin that has the most available capacity, and the available capacity of the bin is updated (the task length is subtracted from the bin capacity).

Step Five: If the task queue is empty, the algorithm terminates. Otherwise, it returns to Step Three.

It is evident that changes need to be made to incorporate the idea of considering RAM. Therefore, we intend to examine these changes precisely.

4.2. *Required Changes to Consider RAM*

As mentioned in previous sections, the main idea of this paper is to include the RAM parameter in the system model. Thus, the algorithm needs to be adjusted to account for RAM. To incorporate RAM into the algorithm, Step Four from the previous section needs to be modified as follows:

Step Four: The i-th task is assigned to the machine whose bin has the most available capacity and whose RAM is greater than the requested RAM for the task RAM_{t_i}. The available capacity of the bin is then updated (the task length is subtracted from the bin capacity).

As we know, modern operating systems have the capability of virtual memory, which utilizes storage memory as virtual RAM through swap operations when RAM is insufficient. However, this introduces more latency compared to using actual RAM. In scenarios where we have idle processing cores but insufficient RAM to execute a task, this feature helps us use the processing cores by accepting the penalty of additional delay, thereby increasing efficiency compared to waiting for RAM to become available. Now, suppose the required RAM for a task exceeds the total RAM of the machine. In this case, the task will always incur the penalty of virtual memory delay, significantly increasing its execution time. However, if we assign this task to a machine with more RAM than required, it will only experience delays when other programs occupy the machine's RAM. By doing this, we only pay the penalty of virtual memory usage when necessary, avoiding unnecessary increases in execution time due to incorrect assignments. Thus, Step Four has been modified accordingly.

5. Implementation

As mentioned, one of the main reasons for considering RAM in this algorithm is the delay associated with using virtual memory. To evaluate this delay in the implementation, we use the CloudSim Plus simulator. This simulator is based on CloudSim but has additional capabilities, such as supporting virtual memory delay, which is effective in implementing the project. Furthermore, two types of heterogeneity (high and low) are considered for the task lengths and computational power of the virtual machines in the simulation. Figure 5-1 illustrates this (Dhahbi et al., 2021).

Table 2: Heterogenous in simulations

		HiHi config
Hosts	Number	Sufficient number to host all VMs
	PEs	4
	MIPS/PE	16k
	Provision policy	Time shared
VMs	Number	20
	PEs	4
	MIPS/PE	1k-16k
	Allocation policy	Space shared
Tasks	PEs	1-4
	Length	50k-350k

The RAM for virtual machines is randomly set within the range of 512 MB to 4 GB, and for tasks, it is within 128 MB to 1792 MB. It is important to note that the allocation policy refers to the virtual machine-to-host allocation policy, and the provisioning policy refers to the CPU allocation policy for each host to virtual machines. In the CloudSim Plus simulator, like in CloudSim, the space-shared algorithm is the same as FCFS (First-Come-First-Serve), and the time-shared algorithm is the same as the round-robin algorithm. Additionally, the scheduling policy, which is the policy for allocating virtual machine cores to tasks, is set to space-shared.

6. Evaluations

In Figures 6-1 and 6-2, the label "wfblba" refers to the results of the algorithm (Dhahbi et al., 2021), and the label "wfblba_n" refers to the modified algorithm without considering virtual memory delay. That is the method we Proposed.

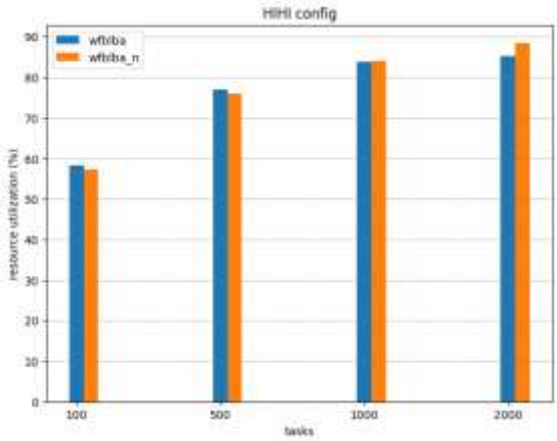

Figure 6-2: The amount of resource utilization in HIHI setting

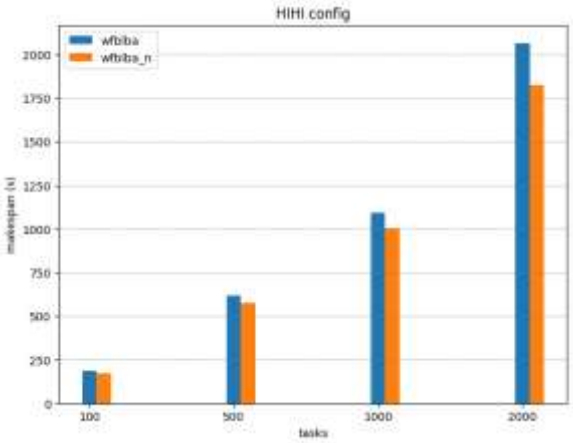

Figure 6-1: The amount of makespan in HIHI setting

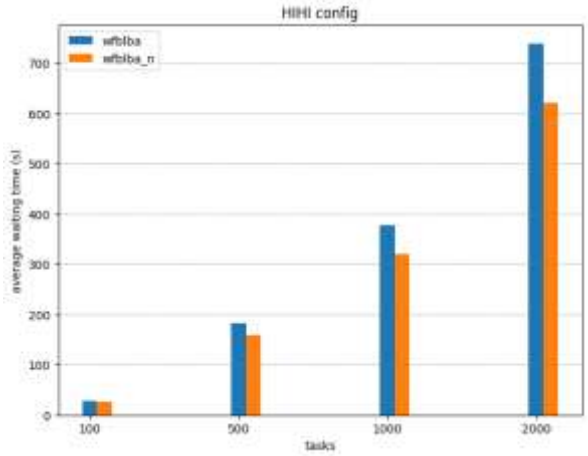

Figure 6-3: Average amount of resource waiting time in the HIHI setting

The interesting point in this section is that resource utilization has increased for a small number of tasks despite considering the virtual memory delay. Figure 6-1 shows makespan. As can be seen, it is clear that due to the delay calculation, makespan increases with the increase in the number of tasks, and as can be seen, this has happened, but it has become less than (Dhahbi et al., 2021) .As can be seen in Figure 6-2, resource utilization should increase with the increase in the number of tasks, and it shows that our proposed

method performs better as the number of tasks increases. Figure 6-3 shows the average waiting time for different numbers of tasks, and as we expected, in our proposed method, as the number of tasks increases, the average waiting time increases, but by comparing our proposed method with task 10, it can be seen that The average waiting time is less. This indicates that our algorithm has reduced the impact of delay by placing tasks on appropriate machines. Another important point is that for a large number of tasks, specifically 2000, the efficiency is worse when delay is considered compared to when it is not. This is because as the number of tasks increases, the impact of virtual memory delay also increases. In large numbers, this impact increases to the extent that even with reduced impact, efficiency will still decrease. It is worth noting that although this article only analyzes three graphs, this finding holds true for other scenarios as well. The proposed method has been able to improve resource efficiency by 20% and also reduced the average waiting time of tasks by 15%, which shows the effectiveness of our proposed method.

7. Conclusion

The increasing complexity and dynamic nature of cloud computing environments necessitate innovative load-balancing strategies that can adapt to fluctuating workloads and resource constraints. In this study, we introduced the Worst-Fit Based Load Balancing Algorithm (WFBLBA), a novel approach that integrates RAM considerations into the load-balancing process to address the critical gap in current methodologies. By modeling the load-balancing problem as a bucket-pulling scenario and employing a worst-fit heuristic, WFBLBA effectively minimizes virtual memory delays and maximizes resource utilization. Our simulation results, conducted using the CloudSim Plus framework, demonstrate the superior performance of WFBLBA over traditional static and dynamic algorithms. Specifically, the WFBLBA achieves a 20% improvement in resource efficiency and a 15% reduction in task processing delays, highlighting its potential to enhance the performance of cloud computing infrastructures significantly. These findings underscore the importance of incorporating memory management into load-balancing algorithms, particularly in environments with diverse and competing resource requirements. The implications of this research extend beyond the immediate performance gains observed in our simulations. By providing a more nuanced understanding of the interplay between task scheduling, memory allocation, and processing delays, WFBLBA offers a robust solution that can adapt to the unique challenges posed by modern cloud environments. This adaptability is crucial for cloud service providers, who must balance the dual objectives of maximizing resource utilization and minimizing latency to meet the needs of a growing and diverse user base. Future research will focus on several key areas to refine and enhance the WFBLBA further. First, we plan to explore the integration of weighted bucket sizes based on task priority and VM capabilities, which could provide an even more precise allocation of resources. Additionally, expanding the simulation framework to include a broader range of cloud configurations and workload scenarios will help validate the algorithm's applicability across different use cases and industry verticals. Another area of interest is the development of advanced tools for accurately simulating resource usage and delays in cloud computing environments. Such tools would enable researchers and practitioners to better understand the complexities of load balancing and task scheduling, ultimately leading to the development of more sophisticated and effective algorithms. In conclusion, the WFBLBA represents a significant advancement in the field of cloud computing, offering a practical and effective means of optimizing task scheduling and resource allocation. By addressing the current shortcomings in load-balancing strategies and paving the way for more efficient cloud computing infrastructures, this research contributes to the ongoing efforts to improve the quality and reliability of cloud services. As cloud computing continues to evolve, the principles and techniques outlined in this study will serve as a foundation for future innovations in load balancing and resource management.

References

Abdulkareem, N. M., Zeebaree, S. R. M., Sadeeq, M. A. M., Ahmed, D. M., Sami, A. S., Zebari, R. R. (2021). IoT and Cloud Computing Issues, Challenges and Opportunities: A Review. Qubahan Academic Journal, 1(2), 1–7. https://doi.org/10.48161/qaj.v1n2a36

Adhikari, M., Amgoth, T. (2018). Heuristic-based load-balancing algorithm for IaaS cloud. Future Generation Computer Systems, 81, 156–165. https://doi.org/10.1016/j.future.2017.10.035

Al-Shareeda, M. A., Alsadhan, A. A., Qasim, H. H., Manickam, S. (2024). The fog computing for internet of things: review, characteristics and challenges, and open issues. Bulletin of Electrical Engineering and Informatics, 13(2), 1080–1089. https://doi.org/10.11591/eei.v13i2.5555

Alsubaei, F. S., Hamed, A. Y., Hassan, M. R., Mohery, M., Elnahary, M. K. (2024). Machine learning approach to optimal task scheduling in cloud communication. Alexandria Engineering Journal, 89, 1–30. https://doi.org/10.1016/j.aej.2024.01.040

Banerjee, S., Adhikari, M., Kar, S., Biswas, U. (2015). Development and Analysis of a New Cloudlet Allocation Strategy for QoS Improvement in Cloud. Arabian Journal for Science and Engineering, 40(5), 1409–1425. https://doi.org/10.1007/s13369-015-1626-9

Devaraj, A. F. S., Elhoseny, M., Dhanasekaran, S., Lydia, E. L., Shankar, K. (2020). Hybridization of firefly and Improved Multi-Objective Particle Swarm Optimization algorithm for energy efficient load balancing in Cloud Computing environments. Journal of Parallel and Distributed Computing, 142, 36–45. https://doi.org/10.1016/j.jpdc.2020.03.022

Dhabliya, D., Dari, S. S., Sakhare, N. N., Dhablia, A. K., Pandey, D., Muniandi, B., George, A. S., Hameed, A. S., Dadheech, P. (2024). New proposed policies and strategies for dynamic load balancing in cloud computing. Emerging Trends in Cloud Computing Analytics, Scalability, and Service Models (pp. 135–143). IGI Global. https://doi.org/10.4018/979-8-3693-0900-1.ch006

Dhahbi, S., Berrima, M., Al-Yarimi, F. A. M. (2021). Load balancing in cloud computing using worst-fit bin-stretching. Cluster Computing, 24(4), 2867–2881. https://doi.org/10.1007/s10586-021-03302-7

Djebbar, E. I., Belalem, G. (2016). Tasks Scheduling and Resource Allocation for High Data Management, Scientific Cloud Computing Environment (pp. 16–27). https://doi.org/10.1007/978-3-319-50463-6_2

Dubey, U., Songare, L. S. (2019). Analysis of load balancing in cloud computing. International Journal of Scientific and Technology Research, 8(12), 3912–3914.

Elsakaan, N., Amroun, K. (2024). A novel multi-level hybrid load balancing and tasks scheduling algorithm for cloud computing environment. Journal of Supercomputing, 80(9), 13434–13474. https://doi.org/10.1007/s11227-024-05990-5

Balaji, K., Sai Kiran, P., Sunil Kumar, M. (2021). Load balancing in Cloud Computing: Issues and Challenges. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(2), 3224–3231. https://doi.org/10.17762/turcomat.v12i2.2380

Geetha, P., Vivekanandan, S. J., Yogitha, R., Jeyalakshmi, M. S. (2024). Optimal load balancing in cloud: Introduction to hybrid optimization algorithm. Expert Systems with Applications, 237, 121450. https://doi.org/10.1016/j.eswa.2023.121450

Gogineni, N., Saravanen, M.S. (2024). Contention-aware Greedy Heuristic Method and Learning-based Method for Load Balancing through Scheduling for Containers in Cloud Computing Environments. Available at SSRN 4775837.

Haris, M., Zubair, S. (2022). Mantaray modified multi-objective Harris hawk optimization algorithm expedites optimal load balancing in cloud computing. Journal of King Saud University - Computer and Information Sciences, 34(10), 9696–9709. https://doi.org/10.1016/j.jksuci.2021.12.003

Jena, U. K., Das, P. K., Kabat, M. R. (2022). Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment. Journal of King Saud University - Computer and Information Sciences, 34(6), 2332–2342. https://doi.org/10.1016/j.jksuci.2020.01.012

Khan, A. R. (2024). Dynamic Load Balancing in Cloud Computing: Optimized RL-Based Clustering with Multi-Objective Optimized Task Scheduling. Processes, 12(3), 519. https://doi.org/10.3390/pr12030519

Khare, S., Chourasia, U., & Deen, A. J. (2022). Load Balancing in Cloud Computing. Cognitive Science and Technology, 601–608. https://doi.org/10.1007/978-981-19-2350-0_58

Kruekaew, B., Kimpan, W. (2022). Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee Colony Algorithm with Reinforcement Learning. IEEE Access, 10, 17803–17818. https://doi.org/10.1109/ACCESS.2022.3149955

Latchoumi, T. P., Parthiban, L. (2022). Quasi Oppositional Dragonfly Algorithm for Load Balancing in Cloud Computing Environment. Wireless Personal Communications, 122(3), 2639–2656. https://doi.org/10.1007/s11277-021-09022-w

Lhomme, A., Catusse, N., Brauner, N. (2024). Computational bounds on randomized algorithms for online bin stretching. ArXiv Preprint ArXiv:2405.19071.

Li, M., Zhang, J., Wan, J., Ren, Y., Zhou, L., Wu, B., Yang, R., Wang, J. (2020). Distributed machine learning load balancing strategy in cloud computing services. Wireless Networks, 26, 16. https://doi.org/10.1007/s11276-019-02042-2

Maheswaran, M., Ali, S., Siegal, H. J., Hensgen, D., Freund, R. F. (1999). Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. Proceedings. Eighth Heterogeneous Computing Workshop (HCW'99), 59, 30–44. https://doi.org/10.1109/HCW.1999.765094

Meera, S., Valarmathi, K. (2024). Optimizing cloud resource allocation: A long short-term memory and DForest-based load balancing approach. Journal of Intelligent and Fuzzy Systems, 46(1), 2311–2330. https://doi.org/10.3233/JIFS-234054

Mishra, R., Gupta, M. (2024). DRABC-LB: A Novel Resource-Aware Load Balancing Algorithm Based on Dynamic Artificial Bee Colony for Dynamic Resource Allocation in Cloud. SN Computer Science, 5(2), 233. https://doi.org/10.1007/s42979-023-02570-x

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. https://doi.org/10.1016/j.jksuci.2018.01.003

Mohanty, S., Patra, P. K., Ray, M., Mohapatra, S. (2021). A Novel Meta-Heuristic Approach for Load Balancing in Cloud Computing. Research Anthology on Architectures, Frameworks, and Integration Strategies for Distributed and Cloud Computing (pp. 504–526). IGI Global. https://doi.org/10.4018/978-1-7998-5339-8.ch023

Muteeh, A., Sardaraz, M., Tahir, M. (2021). MrLBA: multi-resource load balancing algorithm for cloud computing using ant colony optimization. Cluster Computing, 24(4), 3135–3145. https://doi.org/10.1007/s10586-021-03322-3

Neetu Settia, Shivani Duggal, Saksham Kataria. (2024). Unleashing the Potential: Compliance Standards Using Cloud Computing. International Research Journal on Advanced Engineering Hub (IRJAEH), 2(3), 641–644. https://doi.org/10.47392/irjaeh.2024.0092

Negi, S., Rauthan, M. M. S., Vaisla, K. S., Panwar, N. (2021). CMODLB: an efficient load balancing approach in cloud computing environment. Journal of Supercomputing, 77(8), 8787–8839. https://doi.org/10.1007/s11227-020-03601-7

Pradhan, A., Bisoy, S. K. (2022). A novel load balancing technique for cloud computing platform based on PSO. Journal of King Saud University - Computer and Information Sciences, 34(7), 3988–3995. https://doi.org/10.1016/j.jksuci.2020.10.016

Punitha, V., Mala, C. (2021). Traffic classification for efficient load balancing in server cluster using deep learning technique. The Journal of Supercomputing, 77(8), 8038–8062. https://doi.org/10.1007/s11227-020-03613-3

Salehnia, T., Ghaffari, A., Abualigah, L. (2024). Workflow Scheduling in Cloud System: An Optimal Rule-Based Virtual Machine Selection Technique Using MFSSA.

Saravanan, G., Neelakandan, S., Ezhumalai, P., Maurya, S. (2023). Correction: Improved wild horse optimization with levy fight algorithm for effective task scheduling in cloud computing (Journal of Cloud Computing, (2023), 12, 1, (24), 10.1186/s13677-023-00401-1). Journal of Cloud Computing, 12(1), 24. https://doi.org/10.1186/s13677-023-00432-8

Sefati, S. S., Mousavinasab, M., Zareh Farkhady, R. (2022). Load balancing in cloud computing environment using the Grey wolf optimization algorithm based on the reliability: performance evaluation. Journal of Supercomputing, 78(1), 18–42. https://doi.org/10.1007/s11227-021-03810-8

Shafiq, D. A., Jhanjhi, N. Z., Abdullah, A. (2022). Load balancing techniques in cloud computing environment: A review.

Journal of King Saud University - Computer and Information Sciences, 34(7), 3910–3933. https://doi.org/10.1016/j.jksuci.2021.02.007

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. https://doi.org/10.1109/ACCESS.2021.3065308

Sharma, V., Bala, M. (2020). An Improved Task Allocation Strategy in Cloud using Modified K-means Clustering Technique. Egyptian Informatics Journal, 21(4), 201–208. https://doi.org/10.1016/j.eij.2020.02.001

Srivastava, A., Kumar, N. (2024). An efficient firefly and honeybee based load balancing mechanism in cloud infrastructure. Cluster Computing, 27(3), 2805–2827. https://doi.org/10.1007/s10586-023-04118-3

Thakur, A., Goraya, M. S. (2022). RAFL: A hybrid metaheuristic based resource allocation framework for load balancing in cloud computing environment. Simulation Modelling Practice and Theory, 116, 102485. https://doi.org/10.1016/j.simpat.2021.102485

Yu, D., Ma, Z., Wang, R. (2022). Efficient Smart Grid Load Balancing via Fog and Cloud Computing. Mathematical Problems in Engineering, 2022(1), 3151249. https://doi.org/10.1155/2022/3151249