ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 11, Issue 3, pp: (160-171), Month: July - September 2023, Available at: www.researchpublish.com

Comparative Analysis of Load Balancing Algorithms for Efficient Task Scheduling in Cloud Computing

[1]Sawsan Rabaya, [2]Yazeed Al Moayed

Department of Information Technology, Faculty of Computer & Info. Tech

Al-Madinah International University, Kuala Lumpur, Malaysia

DOI: https://doi.org/10.5281/zenodo.8355059

Published Date: 18-September-2023

Abstract: **In the field of information technology cloud computing is a recently developed technology. In such a complicated system, an effective load balancing scheme is critical in order to meet peak user demands and deliver high-quality services. Load balancing is a way of distributing workload among several nodes over network links in order to maximize resource utilization, decrease data processing and response time, and avoid overload. There have been a number of load balancing algorithms suggested that concentrate on important factors including processing time, response time, and processing costs. These techniques, however, ignore cloud computing scenarios. At the same time, there is a few research works that focuses on the subject of load balancing in cloud computing. Motivated by this issue, this study addresses the load balancing challenge in cloud computing by comparing natural inspired Load Balancing Algorithms based on the resource utilization metric. The chosen load balancing methods will next be tested and assessed using the CloudSim simulator to choose the proper natural inspired Load balancing algorithms that solves the problem of load balancing in cloud computing, according to the result of the simulation it can be concluded that the LBA_HB is better than the HBB_LB based on the results for the response time, MakeSpan, and the degree of imbalance.**

Keywords: **Load Balancing, Cloud Computing, Algorithms, CloudSim, HBB-LB, LBA_HB.**

I. INTRODUCTION

The increasing demand for efficient and reliable load balancing algorithms (LBA) has led to the development of various approaches, including those inspired by natural systems. This paper will conduct a performance analysis of two natural inspired Load Balancing Algorithms (NI-LBAs), comparing their effectiveness and efficiency. Nowadays most developments in the information technology [1] industry are driven by the need to use more resources for minimum cost Issawi, Al Halees, and Radi (2015)[2]. This technical trend has facilitated the development of a new computer paradigm known as Cloud Computing (CC), in which resources and services are available on the Internet and can be leased and released on demand [3] CC is defined as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with little management effort or service provider interaction" by The National Institute of Standards and Technology (NIST) [4-6].

LB is a process by which workloads are distributed across multiple computers within a distributed system [7]. This improves the speed of job processing, such as reducing response time (RT) and increasing resource utilization [3] [7]. The primary goal of a LBA is to optimize system performance while keeping costs reasonable [8]. This can include reducing task RT and treating all jobs in the system equally, regardless of their origin [8]. Additionally, LB can improve system fault tolerance (FT), which helps to maintain performance even in the event of partial failures in the system [8].

II. PROBLEM BACKGROUND & PROBLEM STATEMENT

Having a reliable connection is crucial for optimal device performance, especially in a CC environment where resources are both limited and costly. As the number of users and their demands continue to grow at a rapid pace, real-time analysis is becoming increasingly important for certain CC solutions. Users expect to have seamless access to their data from any location, at any time, using any device. This is made possible by storing and processing users' data on cloud servers located at the service provider's site. Furthermore, with the increasing adoption of CC, data security and privacy have become critical issues. It is essential that service providers implement robust security measures such as encryption and multi-factor authentication to ensure the safety of users' data. When it comes to CC, minimizing both data processing time (PT) and RT is key. Latency can also be a significant issue, as it affects the speed at which data is transmitted. Efficient use of resources is necessary to achieve the highest level of productivity. Managing the distribution of workloads across the network can lead to benefits in both performance and cost-effectiveness.

LB is a primary challenge in CC as resources are limited and can be costly. To maximize output, it is crucial to effectively use these resources. While there are several load-balancing algorithms (LBA) that are based on natural-inspired (NI) models, more comprehensive research is needed to compare various natural-inspired load balancing algorithms and determine their efficacy in reducing RT, MS, and VM imbalance. At present, the available researches do not provide a clear understanding of which algorithms are most effective in achieving these goals. Therefore, there is a significant knowledge gap in this area that requires further investigation to develop an optimal load balancing approach for VMs.

III. SELECTED LOAD BALANCING ALGORITHMS

To identify and select the most optimal and efficient algorithm based on the following factors:

A. Natural Inspired Environment

Compared to static and DLBAs, NI-LBAs are designed to mimic the behaviour of natural systems, such as ant colonies, bees, and birds. These algorithms are based on decentralized decision-making and operate in a self-organized manner, adapting to changing conditions and optimizing system performance over time. In contrast, SLBAs typically rely on predefined rules and heuristics to distribute workloads among computing resources. These algorithms are suitable for scenarios where the workload distribution is relatively stable and predictable, but they can struggle to adapt to changing conditions or optimize performance in real-time.

DLBAs, on the other hand, are designed to adjust the workload distribution in real-time based on system conditions and performance metrics. These algorithms are more flexible than static algorithms and can adapt to changing conditions, but they require more computational resources and can introduce additional overhead into the system. While each approach has its strengths and weaknesses, NI-LBAs are particularly effective in natural environments where workload distribution can be highly variable and unpredictable. By leveraging the self-organizing and adaptive properties of natural systems, these algorithms can optimize performance and minimize the impact of environmental factors on system performance.

B. Resource Utilization Metric as Basic Metric:

This research also filters the remaining data based on RU metric. RU is the degree to which computing resources, such as CPU, memory, and storage, are being used to complete tasks. It is a critical metric that reflects the efficiency of resource allocation and utilization in distributed computing systems. By filtering based on RU, it can identify LBAs that are designed to optimize RU and minimize resource waste. This can significantly improve the performance and efficiency of distributed computing systems, especially in scenarios where resources are limited or expensive. Overall, filtering based on RU is an important step in the research that can help to identify the most resource-efficient LBAs and optimize system performance.

C. The Algorithms with the Largest Number of Performance Metrics:

Refine the information and select the best performing LBA. The filter was designed to choose the algorithms with the maximum number of metrics, eliminating irrelevant or redundant data to improve the accuracy and specificity of our dataset. This allowed us to make more accurate and specific conclusions about the performance and applicability of different load balancing algorithms. Moreover, resulting dataset that is more accurate and specific than the original dataset. This has significant implications for the field of distributed computing, as our findings can inform the design and implementation of LBAs that are more efficient and effective in real-world scenarios.

During this stage of the filtering process, the researcher will conduct a more in-depth analysis of the algorithms. This analysis will involve a closer examination of the individual algorithms and their specific characteristics, allowing for a more detailed understanding of their performance and potential applications.

LBA_HB: The Load Balancing Algorithm Honey Bee (LBA_HB) assigns tasks to VMs based on a combination of the current job count and priority values, avoiding overloading. It follows the behaviour of honey bees (HB). When a task is assigned, the status of the VM is updated and other tasks are notified. Tasks are queued following the First Come First Serve (FCFS) principle, and are assigned to the VM with the lowest workload. If no VM is available, the task will wait for the next available one [9].

HBB-LB: The Honey Bee behaviour Load Balancing (HBB-LB) algorithm is designed to make task assignment more efficient by dividing the public cloud into three categories based on the workload of each VM. The categories include under loaded, balanced, and overloaded. The algorithm then calculates the difference in workload between VMs and, if the difference is greater than zero, it determines there's no need to reassign tasks between VMs. To further optimize the task assignment process, the algorithm takes into account the speed and cost of the VMs when prioritizing tasks. By combining these factors, the HBB-LB algorithm reduces the time needed to find the best VM for each task, improving overall efficiency [10].

HB+RR: In [1], the authors propose a combination of the HB and Round Robin (RR) algorithms as a LB solution. This hybrid approach, called HB+RR, is inspired by the behaviour of bees and aims to improve LB efficiency. The authors argue that while the HB algorithm addresses the issue of prioritization, it still has limitations when handling a high number of tasks. The integration of the RR algorithm aims to address this limitation, however, the static quantum used in RR can still raise issues when handling a high number of tasks.

D. Finalizing and Filtering

The research findings indicate that both HBB-LB and LBA_HB are LBAs that are entirely based on natural inspiration, specifically the behaviour of honeybees. On the other hand, the HB+RR algorithm is a blend of techniques, it combines the Round Robin static algorithm with inspiration from honeybee behaviour. However, in this research, the HB+RR algorithm is not considered, since the research focuses on pure NI and omits hybrid LB techniques.

IV. EXPERMINETS

the VM LBA is implemented using the following software's and tools: Windows 11 Operating System, Apache NetBeans IDE 13, Java Development Kit (JDK) 1.8 and Cloud Sim 3.0.3 tool. These algorithms are implemented for IaaS model in a simulated cloud environment. In this chapter, the research performed a thorough analysis of the performance of LBA_HB and HBB-LB.

This experiment aims to evaluate the performance of the selected NI-LBAs in normal workload conditions. LBA_HB and HBB-LB will be tested using configurations for normal workloads in a homogeneous environment, where the DC has the same number of CPUs and the same CPU speed. The results of this experiment will provide insight into how well these LBAs perform under normal workload conditions in a homogeneous environment.

Table 1 present the parameters used in the simulation of LBA_HB and HBB-LB. The simulation includes 10 DCs, each with 2-6 hosts. The scheduler for VMs is set as time shared and the host specifications include a processing speed of 10,000 MIPS, storage of 100,000 megabyte (MB), bandwidth of 80,000 Mbps, and RAM of 4096 MB. The simulation also includes 50 VMs with varying processor speeds between 500-2000 MIPS, available memory space of 256-2048 MB, and bandwidth of 500-1000. The scheduler for cloudlets is set as time shared and the number of required processor elements for tasks ranges from 1-4. The VM manager used is Xen. The tasks or cloudlets have a length of 1000-20000 bytes and a total of 100-1000 tasks are executed in the simulation.

TABLE I: SIMULATOR PARAMETERS

Type	Parameters	Values
DC	Number of DC	10
	Number of Hosts	2-6
	VM Scheduler	Time shared
	Host Processing speed (MIPS)	10000

	Host Storage (MB)	100000
	Host Bandwidth (Mbps)	80000
	Host Ram (MB)	4096Mb
VM	Number of VMs	50
	Processing speed	500-2000MIPS
	Available memory in single VM	256-2048 Mb
	Bandwidth	500-1000
	Cloudlet Scheduler	Time shared
	Number of processing elements (PEs) request	1-4
	VM manager	Xen
Task or Cloudlet	Length of task (Executable instruction length in bytes)	1000-20000
	Total number of tasks	100-1000

V. RESULTS

LBA are designed to optimize resource utilization, minimize response time, and improve system performance by distributing workloads across multiple computing resources in a network. To assess the performance of HBB-LB and LBA-HB, three crucial metrics were used: RT, MS, and DI. Response time measures the time it takes for a system to respond to a user request, while MS is a measure of the total time required to complete a set of tasks. DI refers to the uneven distribution of workload across multiple computing resources. The results of the experiment are detailed as described below:

A. Response Time:

Next sections show a comparison of the RTs for the LBA-HB and HBB-LB algorithms, giving a clear view of how these two methods perform in terms of this specific metric. By analysing the data, one can determine which algorithm has a better RT and make an informed decision on which method to use in a particular situation.

a) Minimum Response Time

As illustrated in Fig. 1, the Minimum RT for HBB-LB and LBA_HB are presented. The graph clearly demonstrates that LBA_HB has a lower Minimum RT in comparison to HBB-LB, indicating that LBA_HB has a faster RT. This implies that LBA_HB is able to process and respond to requests more efficiently than HBB_LB.

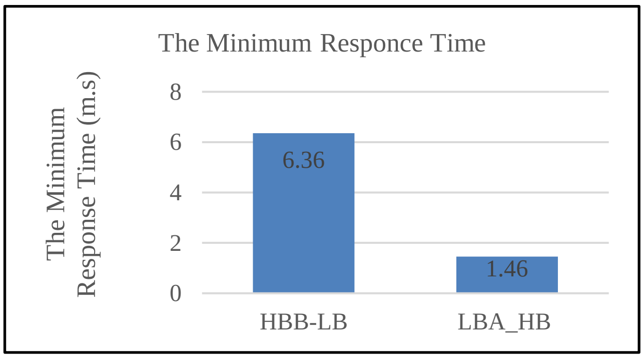

Fig. 1 The Minimum Response Time

b) The First Quartile Response Time

Fig. 2, illustrate the First Quartile RT for both HBB-LB and LBA-HB. The results show that LBA-HB has a lower First Quartile RT compared to HBB-LB. This suggests that LBA-HB is able to process requests faster and return responses in a shorter amount of time than HBB-LB.

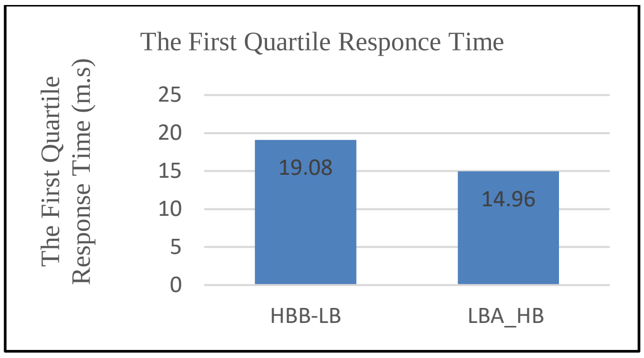

Fig. 2 The First Quartile Response Time

c) The Median Response Time

Fig. 3 illustrates the Median RT for both HBB-LB and LBA-HB. The results show that LBA-HB has a lower median RT compared to HBB-LB. This suggests that LBA-HB is more consistent in processing requests and returning responses in a shorter amount of time than HBB-LB.

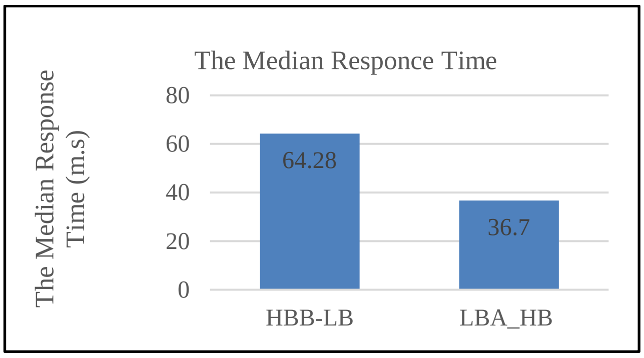

Fig. 3 The Median Response Time

d) The Mean Response Time

Fig. 4, illustrates the Mean RT for both HBB-LB and LBA-HB. The results show that LBA-HB has a lower mean RT compared to HBB-LB. This suggests that on average, LBA-HB is more efficient in processing requests and returning responses in a shorter amount of time than HBB-LB.

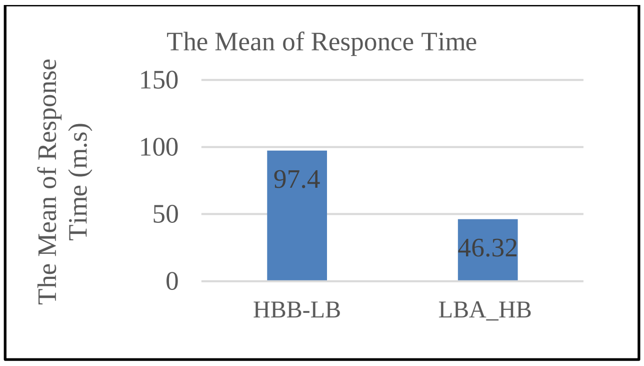

Fig. 4 The Mean of Response Time

e) The Third Quartile Response Time

Fig. 5, illustrates the third Quartile RT for both HBB-LB and LBA-HB. The results show that LBA-HB has a higher third Quartile RT compared to HBB-LB. This suggests that LBA-HB is less consistent in processing requests and returning responses in a shorter amount of time than HBB-LB.

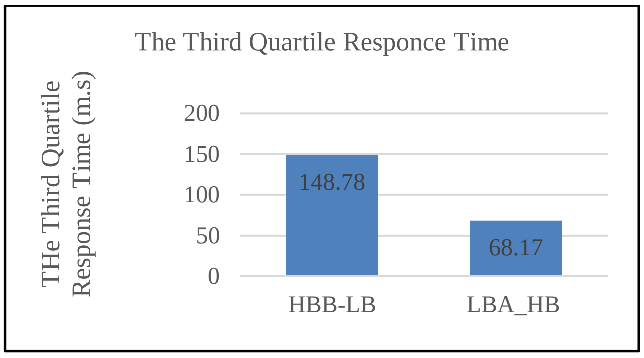

Fig. 5 The Third Quartile Response Time

f) The Maximum Response Time

Fig. 6, illustrates the Maximum RT for both HBB-LB and LBA-HB. The results show that LBA-HB has a lower Maximum RT compared to HBB-LB. This suggests that LBA-HB is more efficient in handling the most complex and longest requests, taking less time than HBB-LB to complete them.

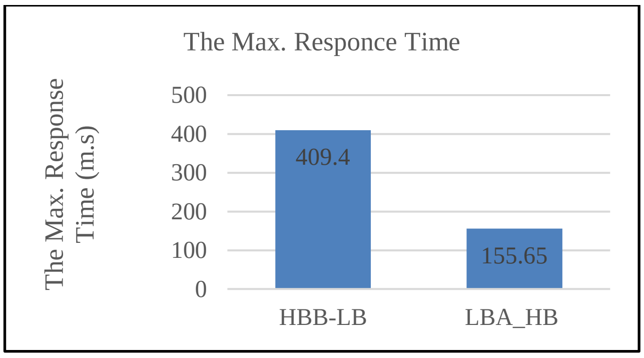

Fig. 6 The Max Response Time

B. MakeSpan

The following sections shows a comparison of the results obtained from the LBA-HB and HBB-LB algorithms in terms of MS. It compares the performance of the two algorithms and highlights any significant differences in the results.

a) The Minimum MakeSpan

Fig. 7 illustrates the comparison of the Min MS between the HBB-LB and LBA_HB algorithms. The figure clearly shows that the LBA_HB algorithm has a lower Min MS compared to the HBB-LB algorithm. This indicates that the LBA_HB algorithm is more efficient in terms of minimizing the CT of a task as compared to the HBB-LB algorithm.

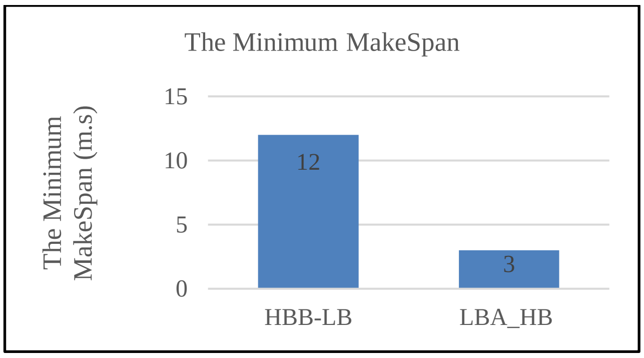

Fig. 7 The Minimum MakeSpan

b) The First Quartile MakeSpan

Fig. 8 represents the comparison of the first quartile MS for the HBB-LB and LBA_HB algorithms. The figure clearly demonstrates that the LBA_HB algorithm has the minimum first quartile MS when compared to HBB_LB. This indicates that LBA_HB is more efficient in terms of time management and completing tasks within a shorter time frame.

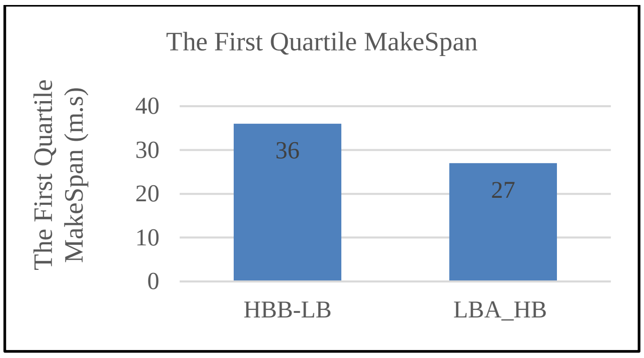

Fig. 8 The First Quartile MakeSpan

c) The Median MakeSpan

Fig. 9, presents a comparison of the median MS for the HBB-LB and LBA_HB algorithms. The figure demonstrates that the LBA_HB algorithm has a lower median MS compared to HBB_LB. This suggests that LBA_HB is more efficient in terms of time management and completing tasks within a shorter time frame.

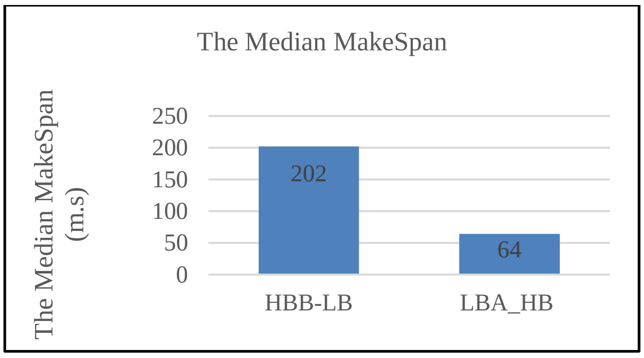

Fig. 9 The Median MakeSpan

d) The Mean MakeSpan

Fig. 10, compares the average MS for the HBB-LB and LBA_HB algorithms and shows that LBA_HB has a shorter average MS than HBB_LB. This means that LBA_HB is more efficient in completing tasks in a shorter amount of time. Furthermore, since the mean is a widely used measure of central tendency, it suggests that LBA_HB is well-suited for time-sensitive situations where minimizing the MS is of the utmost importance and overall performance is a key consideration.

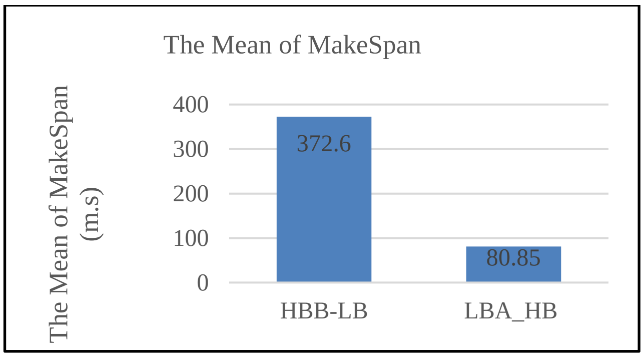

Fig. 10 The Mean of MakeSpan

e) The Third Quartile MakeSpan

Fig. 11, illustrates the comparison of the third quartile MS for the HBB-LB and LBA_HB algorithms. The figure clearly demonstrates that the HBB-LB algorithm has the maximum third quartile MS when compared to LBA_HB. This indicates that HBB-LB is less efficient in terms of time management and completing tasks within a shorter time frame. Additionally, since the third quartile is the highest point of a dataset, this result suggests that HBB-LB is not well-suited for real-time or time-sensitive applications where minimizing the MS is crucial. Furthermore, this comparison also suggests that LBA_HB may have a better performance in terms of time management and CT in the upper range of the dataset.

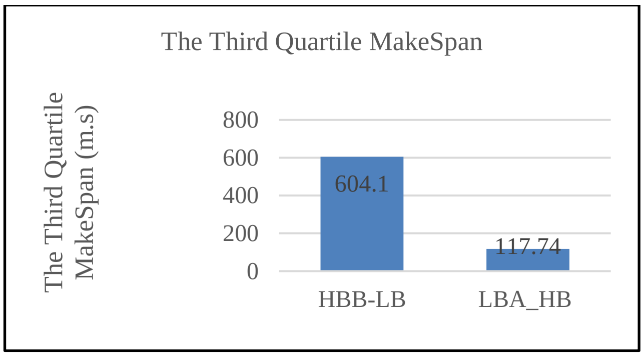

Fig. 11 The Third Quartile MakeSpan

f) The Maximum MakeSpan

Fig. 12, illustrates the comparison of the maximum MS for the HBB-LB and LBA_HB algorithms. The figure clearly demonstrates that the LBA_HB algorithm has a lower maximum MS when compared to HBB_LB. This suggests that LBA_HB is more efficient in terms of time management and completing tasks within a shorter time frame, even in the cases where the task takes longer to complete. Additionally, this comparison also suggests that LBA_HB may have a better performance in terms of time management and CT even in the worst-case scenario.

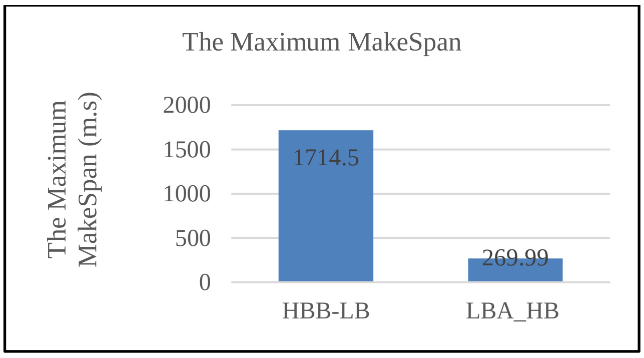

Fig. 12 The Maximum MakeSpan

C. The Degree of Imbalance

By analysing the data in the next sections, one can determine which algorithm has a better performance in terms of DI and make an informed decision on which method to use in a particular situation, or which algorithm is more suitable in terms of LB. This information can be useful in optimizing the performance of a CC system and ensuring that resources are distributed efficiently.

a) The Minimum Degree of Imbalance

Fig. 13 provides a comprehensive comparison of DI between the LBA-HB and HBB-LB algorithms. The figure presents the results of the two methods in terms of the minimum DI. The figure illustrates that the HBB-LB algorithm has a higher minimum DI compared to the LBA-HB algorithm, which suggests that the latter may have a more efficient and balanced distribution of resources among the DCs. This information can be useful for system administrators to make an informed decision on which algorithm to use in their specific system.

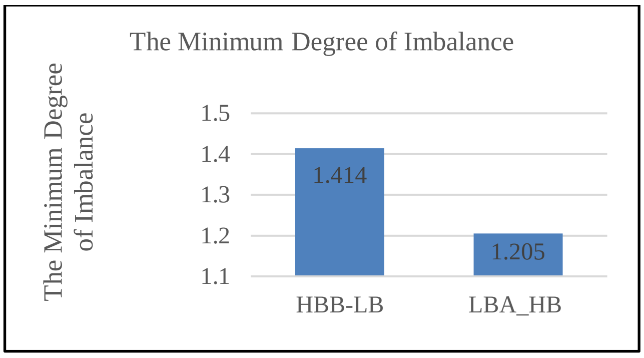

Fig. 13 The Minimum Degree of Imbalance

b) The First Quartile Degree of Imbalance

Fig. 14, illustrates the first quartile of DI for the HBB-LB and LBA_HB algorithms. The graph demonstrates that HBB-LB has a higher DI than LBA_HB, indicating that the former algorithm is less balanced in terms of resource distribution among the VMs.

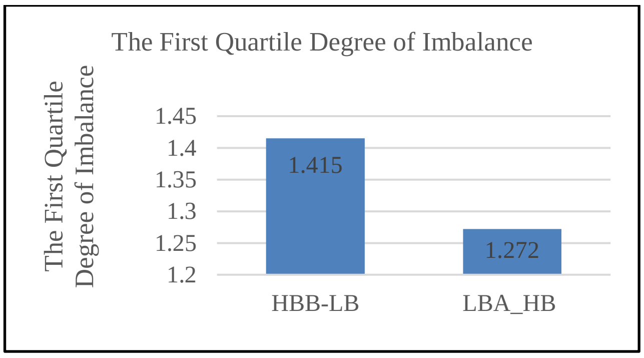

Fig. 14 The first Quartile Degree of Imbalance

c) The Median Degree of Imbalance

Fig. 15, illustrates the median DI for the HBB-LB and LBA_HB algorithms. The graph demonstrates that LBA_HB has a lower median DI than HBB_LB, indicating that the former algorithm is more balanced in terms of resource distribution among the VMs.

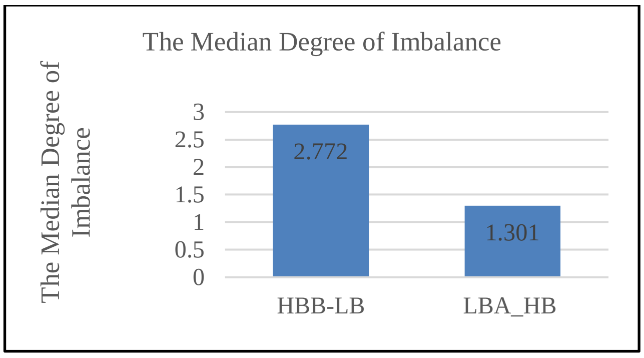

Fig. 15 The Median Degree of Imbalance

d) The Mean Degree of Imbalance

Fig. 16, illustrates the mean DI for the HBB-LB and LBA_HB algorithms. The graph demonstrates that HBB-LB has a higher DI compared to LBA_HB, indicating that the former algorithm does not distribute resources as evenly among the VMs as the latter algorithm does.

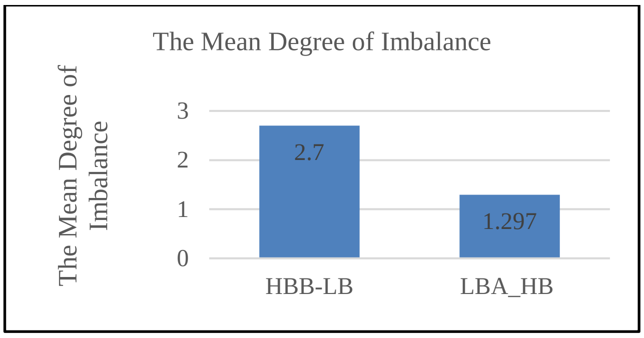

Fig. 16 The Mean Degree of Imbalance

e) The Third Quartile Degree of Imbalance

Fig. 17, illustrates the comparison of the third quartile of DI between the HBB-LB and LBA_HB algorithms. The graph demonstrates that HBB-LB has a higher DI than LBA_HB, which implies that LBA_HB is more balanced in terms of resource distribution among the VMs. This can have a significant impact on the overall performance and efficiency of the system, and it's important to consider when choosing LBA.

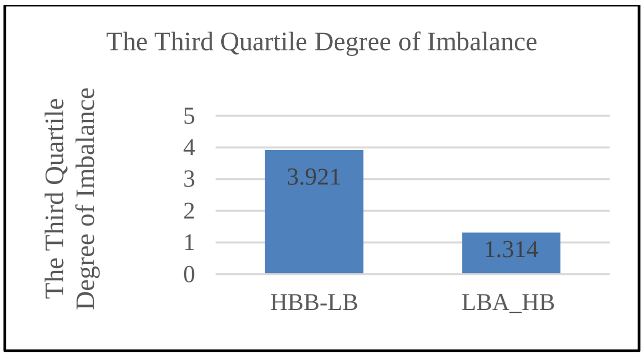

Fig. 17 The Third Quartile Degree of Imbalance

f) The Maximum Degree of Imbalance

Fig. 18, compares the maximum DI for the HBB-LB and LBA_HB algorithms. The graph demonstrates that HBB-LB has a higher maximum DI compared to LBA_HB, indicating that the former algorithm may not be as efficient in terms of balancing resources among the VMs.

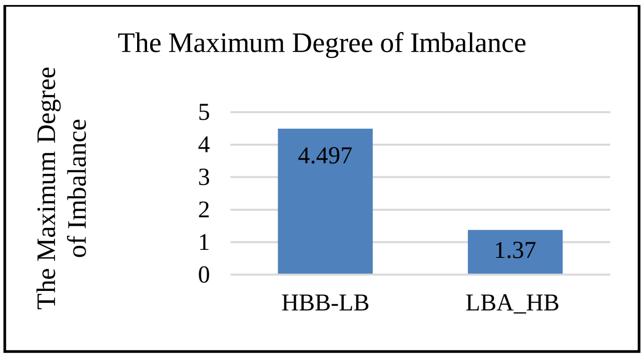

Fig. 18 The Maximum Degree of Imbalance

VI. DISCUSSION

The analysis provides insights into the effectiveness of these algorithms in improving system performance and achieving LB. It compares the minimum RT, first quartile RT, median RT, mean RT, third quartile RT, and maximum RT 0for HBB-LB and LBA_HB. Similarly, it compares the minimum MS, first quartile MS, median MS, mean MS, third quartile MS, and maximum MS for both algorithms. In addition, it examines the DI for HBB-LB and LBA_HB, as this metric provides insight into the extent to which the workload is evenly distributed across processors. Based on the results presented, it can be concluded that the LBA_HB algorithm outperforms the HBB-LB algorithm in terms of RT and MS. The LBA_HB algorithm has a significantly lower minimum RT of 1.46 ms, compared to HBB-LB's minimum RT of 6.36 ms. Similarly, LBA_HB has a lower 1st Quartile RT, median RT, and third quartile RT than HBB-LB. This indicates that LBA_HB provides faster RT on average and has a narrower distribution of RTs.

The MS results further reinforce the superiority of LBA_HB over HBB-LB. LBA_HB has a minimum MS of 3.00 ms, compared to HBB-LB's minimum MS of 12.0 ms. LBA_HB also has lower 1st Quartile, median, and third quartile MS values than HBB-LB, indicating that it is more efficient at completing tasks within a given time frame. However, it is worth noting that the HBB-LB algorithm has a higher maximum RT and MS than LBA_HB, indicating that HBB-LB may struggle with particularly challenging tasks. Additionally, DI is generally higher for HBB-LB than for LBA_HB, which may indicate that HBB-LB is more susceptible to workload imbalances.

Based on the results, LBA_HB outperformed HBB-LB in terms of RT and MS metrics, indicating that it may be more suitable for handling fewer complex tasks. However, HBB-LB may be better suited for handling more complex or challenging tasks, and DI may impact its performance when dealing with varying workloads. These findings have implications for the design and implementation of LBAs in distributed computing environments, enabling system administrators to make informed decisions about which algorithm to use in a given environment to improve system performance and achieve LB.

VII. CONTRIBUTION

The researcher focused on LB in CC and its importance in improving performance and reducing problems and organizing work. The research first provides background information on LBAs and the problems associated with them in CC. A literature review is then presented, providing an overview of CC, its architecture and service models, and the LB model and metrics used to evaluate performance. The research then focuses on the selection of LBAs and the use of simulation tools such as CloudSim. The study also describes and compares the LBA_HB and HBB-LB algorithms in terms of their performance and efficiency using CloudSim simulator. The research analysis provides insights into the effectiveness of these algorithms, comparing minimum, first quartile, median, mean, third quartile, and maximum RT and MS for both algorithms, as well as examining the DI. The results have implications for LBA design and implementation in distributed computing environments, enabling system administrators to make informed decisions to improve system performance and achieve LB.

VIII. CONCLUSION

In conclusion, our study explored the essential role of LB in distributed computing environments, focusing on optimizing task scheduling for efficient processor allocation. We evaluated two LB algorithms, HBB-LB and LBA_HB, using response time (RT), MakeSpan (MS), and workload distribution uniformity (DI) as performance measures. Through comprehensive analysis, we found that LBA_HB outperforms HBB-LB in terms of RT and MS. LBA_HB demonstrated notably lower minimum RT and MS values, indicating faster average response times and more efficient task completion. The results also suggest that LBA_HB maintains a more consistent distribution of task timings.

However, it's important to note that HBB-LB showed higher maximum RT and MS values, possibly indicating its capacity to handle complex tasks. Additionally, HBB-LB's higher DI values hint at its potential susceptibility to workload imbalances. In summary, LBA_HB excels in less complex scenarios, offering quicker response times and more efficient task completion. On the other hand, HBB-LB could prove advantageous for intricate tasks, though it might face challenges with extreme cases and varying workloads. These insights guide the selection of LB algorithms in distributed computing setups, empowering administrators to optimize system performance and achieve load balancing goals.

REFERENCES

[1] Kiritbhai, P.B. and N.Y. Shah, *Optimizing Load Balancing Technique for Efficient Load Balancing.* Int. J. Innov. Res. Technol, 2017. **4**(6): p. 39-44.

[2] Issawi, S.F., A. Al Halees, and M. Radi, *An efficient adaptive load balancing algorithm for cloud computing under Bursty workloads.* Engineering, Technology & Applied Science Research, 2015. **5**(3): p. 795-800.

[3] Surianarayanan, C. and P.R. Chelliah, *Essentials of Cloud Computing*. 2019, Springer.

[4] Mishra, R. and A. Jaiswal, *Ant colony optimization: A solution of load balancing in cloud.* International Journal of Web & Semantic Technology, 2012. **3**(2): p. 33.

[5] Mohammadian, V., et al., *LBAA: A novel load balancing mechanism in cloud environments using ant colony optimization and artificial bee colony algorithms.* International Journal of Communication Systems, 2023. **36**(9): p. e5481.

[6] Ijeoma, C.C., et al., *Review of Hybrid Load Balancing Algorithms in Cloud Computing Environment.* arXiv preprint arXiv:2202.13181, 2022.

[7] Agrawal, N., *Dynamic load balancing assisted optimized access control mechanism for edge‐fog‐cloud network in Internet of Things environment.* Concurrency and Computation: Practice and Experience, 2021. **33**(21): p. e6440.

[8] Gauhar Fatima, S., et al., *Cloud computing and load balancing.* International Journal of Advanced Research in Engineering and Technology, 2019. **10**(2): p. 189-209.

[9] Hashem, W., H. Nashaat, and R. Rizk, *Honey bee based load balancing in cloud computing.* KSII Transactions on Internet and Information Systems (TIIS), 2017. **11**(12): p. 5694-5711.

[10] Ehsanimoghadam, P. and M. Effatparvar, *Load balancing based on bee colony algorithm with partitioning of public clouds.* International Journal of Advanced Computer Science and Applications, 2018. **9**(4).