Smart Internet of Things

www.siot.reapress.com

Smart. Internet. Things. Vol. 1, No. 1 (2024) 17–30.

Paper Type: Original Article

Critical Analysis on Challenges and Strategies of Load Balancing in Hybrid Cloud

Akash Kumar[1],*, Mohit Tiwary[1], Ritwik Dev Singh[1]

[1] Department of Computer Engineering, KIIT University, Bhubaneswar-751024, Odisha, India; 21051624@kiit.ac.in; 21051314@kiit.ac.in; 21051331@kiit.ac.in.

Citation:

Received: 4 May 2024 *Revised: 14 July 2024* *Accepted: 11 August 2024*	Kumar, A., Tiwary, M., Dev Singh, R. (2024). Critical analysis on challenges and strategies of load balancing in hybrid cloud. *Smart Internet of Things, 1* (1), 17-30.

Abstract

Cloud computing has changed how organizations handle their IT, with many now using hybrid clouds that mix private and public cloud resources. This shift has led to big changes in how IT is managed. Hybrid clouds, which blend private and public cloud resources, are now very common among businesses. In these setups, load balancing is crucial for spreading tasks across different cloud services effectively. This helps use resources better and make systems work more efficiently. This research paper explores the challenges and strategies related to load balancing in hybrid cloud environments. This paper investigates these challenges and offers strategies to overcome them, aiming to provide insights for effectively managing workloads across diverse cloud infrastructures. Here we show, how organizations continue to adopt hybrid cloud environments, therefore load balancing remains a crucial aspect of managing workloads across diverse cloud infrastructures. Understanding the challenges and intricacies involved in load balancing can aid organizations in optimizing resource utilization and enhancing overall system efficiency in their hybrid cloud deployments.

Keywords: Hybrid cloud, Cloud computing, Scalability, Performance optimization, Dynamic workload, Virtualization, Auto scaling.

1 | Introduction

Cloud computing has changed how organizations manage their IT, and many now use hybrid clouds, combining private and public cloud resources. Organizations have witnessed a significant transformation in IT management with the adoption of cloud computing [1]. The prevalent use of hybrid clouds, which involve the integration of both private and public cloud resources, has become a common practice among many enterprises. In these hybrid setups, load balancing plays a critical role in ensuring the efficient distribution of

Corresponding Author: 21051624@kiit.ac.in

https://doi.org/10.22105/siot.v1i1.33

Licensee System Analytics. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

tasks across various cloud services [2]. This strategic distribution is aimed at optimizing resource utilization and enhancing overall system efficiency. This paper delves into the intricate challenges associated with load balancing in hybrid cloud environments [3]. By investigating these challenges, the paper aims to provide valuable insights into the complexities that organizations face when managing workloads across diverse cloud infrastructures [2]. We have three main ways of using cloud services, and they are called Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS).

I. Infrastructure-as-a-Service (IaaS): IaaS provides IT resources like servers and storage as a service. You can use and control these resources through a cloud interface. Examples: AWS, Rackspace, Metacloud by Cisco, and others offer IaaS [1–4].

II. Platform-as-a-Service (PaaS): PaaS gives you a ready-made platform with pre-set IT resources that are ready to use. You can use this platform without worrying about setting up or configuring things. Examples: Windows Azure, Google App Engine, and similar platforms are examples of PaaS [4].

III. Software-as-a-Service (SaaS): SaaS provides programs and data as a service to users. Users can use these programs and access data without having to install anything. Examples: Google Apps, Cisco WebEx, Salesforce.com, and others are SaaS applications. So, these delivery models give different ways for people and businesses to use and benefit from cloud services [1], [4].

IV. Additionally, the paper proposes and discusses various strategies to address these challenges effectively. As organizations increasingly shift towards the adoption of hybrid cloud architectures, the need for a comprehensive understanding and successful implementation of efficient load balancing mechanisms becomes imperative. This is crucial for achieving optimal performance, scalability, and resource utilization in the dynamic and complex landscape of hybrid cloud computing [4].

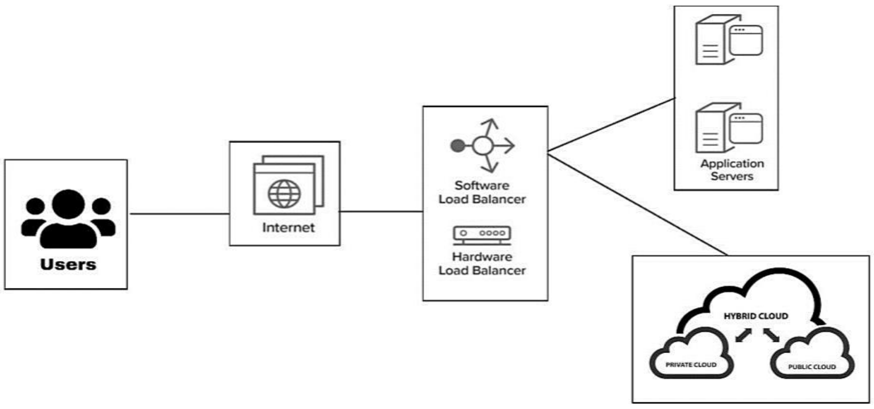

Fig. 1. Illustrate load balancing in hybrid cloud computing.

Two main types of load balancing methods:

I. Static cloud load balancing: static load balancing strategies, such as FIFO, Round Robin, Min-Max, and Max-Min, don't consider the current status of the system. Instead, they rely on predetermined factors like operation time, storage, memory space, and module capabilities. These strategies are easy to implement and execute, making them suitable for small-scale systems. However, they are impractical for estimating system performance and don't facilitate resource distribution during processing. In a static load balancing method, all job-related data is collected in advance to reduce waiting time. Tasks are prioritized based on their processing time, with shorter tasks being executed first and longer tasks later. This approach can help mitigate starvation issues by ensuring that tasks requiring more processing power aren't continuously delayed [3–5].

II. Dynamic cloud load balancing: now, let's consider a different scenario for dynamic load balancing, still using the dinner party analogy. With dynamic load balancing, you're flexible and adjust the seating arrangement based on what's happening in real-time during the party. For instance, if more guests arrive unexpectedly, you rearrange the seating to accommodate everyone comfortably. Similarly, if some guests leave early, you may consolidate the remaining guests to free up space. The key is to adapt and make changes as needed to ensure everyone has a good experience, even if it means adjusting the plan on the fly [6], [7].

Load balancing model

Cloud technology provides users access to resources like physical servers, virtual servers, memory, and networks as needed. To handle user requests effectively, a good load balancing algorithm is essential. Here's how it works.

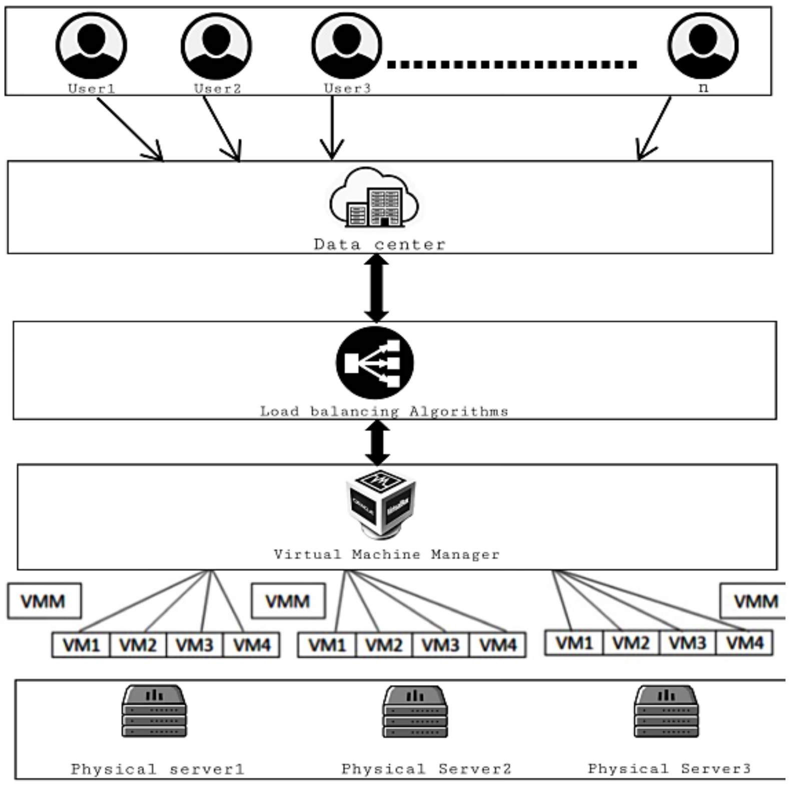

Fig. 2. Load balancing model.

The cloud services receive various requests from users, requiring a dynamic environment to process them efficiently. When a request comes in, the load balancer selects the appropriate Virtual Machines (VMs) based on the load balancing algorithms [8]. The load balancer's job is to distribute user requests among different VMs. The Data Center Controller (DCC) manages job management, forwarding tasks to the load balancer, which then assigns them to suitable VMs. VMs are managed by the VMs manager and are responsible for running users' applications and services. Since customers come from different places and submit requests randomly, job allocation becomes critical in cloud computing. If some VMs are overloaded while others are underutilized, it can affect the Quality of Service (QoS). This can lead to unsatisfied users who may switch to other cloud providers [9]. Therefore, maintaining a good load balancing algorithm is crucial for optimal service performance, ensuring that resources are allocated efficiently to meet users' needs [10].

Strategies of load balancing

Load balancing is a critical aspect of distributed systems, ensuring optimal resource utilization, maximizing throughput, minimizing response time, and avoiding overload on any individual component. Various metrics are used to evaluate the effectiveness of load balancing algorithms and strategies [11]. Some of the key metrics include:

I. Performance: this is about how well the system works once a particular load balancing technique is put in place. We need to make sure that the system performs effectively [12].

II. Makespan: think of makespan as the total time it takes to get everything done, from assigning resources to users to completing their tasks.

III. Throughput: throughput is how many requests the system can handle in a given amount of time. The higher the throughput, the better the system is performing [13].

IV. Scalability: scalability measures how well the system can handle more users or workload without breaking down. It's important that the system can keep balancing the load even as more demand comes in.

V. Fault tolerance: this is about how well the system can keep working even if something goes wrong, like if a server crashes or a connection fails [14].

VI. Response time: response time is how long it takes for the system to respond to a request. We want this to be as short as possible for a better user experience.

VII. Migration time: migration time is the time it takes to move a request from one overloaded machine to another that can handle it better. We want this time to be short to keep things running smoothly [9].

VIII. Resource use: we want to make sure that all the resources available in the system are being used effectively. This helps keep costs down and reduces energy usage.

IX. Degree of imbalance: this measures how evenly the workload is distributed among different parts of the system, like VMs. We want to minimize any big differences to keep things running smoothly.

2 | Literature Review

Researchers worldwide are continuously developing various methods to achieve this balance dynamically during runtime. Aside from load balancing, cloud computing faces other challenges such as VMs migration, execution time, VM performance, energy efficiency, carbon emissions, QoS, and resource management [6–8]. Researcher have explored heuristic-based algorithms to optimize resource allocation in cloud environments by considering factors like network, CPU, and memory loads [10]. Other researchers focus on resource allocation for handling large job requests efficiently [11]. They have discussed resource scheduling and pricing optimization, achieving better success rates with algorithms like MQLO and have studied various task scheduling techniques and identified suitable measures for cloud environments [12–14]. Additionally, some authors have addressed security measures in load balancing environments [14, 15]. The objectives of this article include reviewing existing Cloud Load Balancing (CLB) algorithms, introducing new CLB algorithms, and analyzing their advantages and disadvantages. Load balancing in cloud computing is a crucial task that ensures efficient utilization of resources and smooth operation of services.

The research by Mishra et al. [15] explored heuristic-based algorithms to optimize performance in cloud environments by applying various types of loads such as network, CPU, and memory. They aimed to enhance performance by strategically allocating resources based on different types of loads. Balaji and Saikiran [5] addressed resource allocation problems by providing optimal solutions for handling large job requests efficiently. Their work focused on effectively allocating resources to meet the demands of numerous job requests. Radha et al. [14] discussed resource scheduling in cloud environments and optimized resource provisioning costs while improving the average success rate using the MQLO algorithm. Arunarani et al. [9] conducted a comprehensive study on various task scheduling techniques suitable for cloud environments. They identified and evaluated measures appropriate for optimizing task scheduling based on different methods, applications, and parameters [16]. Several authors [9], [17] emphasized security measures in load balancing environments, integrating security considerations with different metrics to ensure a secure and efficient load balancing process. Parsa and Entezari-Maleki [18] proposed the RASA (Resource Aware Scheduling Algorithm), which is designed for grid environments. RASA combines the advantages of well-known task scheduling algorithms, Min-min and Max-min, while addressing their limitations. The algorithm allocates resources to tasks based on the number of available resources, employing a combination of Min-min and Max-min procedures to allocate resources efficiently. Simulation results demonstrated that RASA outperformed conventional algorithms in terms of performance [18].

3 | Proposed Work

This section contains the working of the proposed model which states that load balancing in hybrid cloud can be achieved through few algorithms which we will discuss later in this section. It is very important to manage the workload in a computer system, especially when the demand for tasks can change suddenly, and different parts of the system can handle different amounts of tasks. As we have discussed earlier, there are two main ways to balance the workload: static and dynamic. Now let's talk about the hybrid load balancing algorithms [19]. Hybrid load balancing algorithms–Algorithms which are achieved and are proposed to overcome the limitations of static and dynamic load balancing method by aggregating the advantages of both the load balancing methods. Instead of using either one of the algorithms, new algorithms are being developed which takes the best parts of both the algorithms. In simpler words, combining the strengths of different existing algorithms, whether they're dynamic or static, and blending them together to make a better and optimized algorithm [20].

3.1 | Genetic Algorithm and Fuzzy Theory

A hybrid job scheduling algorithm is proposed which uses genetic algorithm and fuzzy theory. The genetic algorithm is like a computerized version of evolution, to figure out the best way to assign jobs to resources. The algorithm assigns jobs to resources without considering the job length and resource capabilities. This algorithm was then optimized with the addition of fuzzy theory, which helps deal with uncertainty to make the process more efficient. Different types of job assignments are created, each with its own QoS parameters, and then evaluated how good they were. By using this hybrid approach, the standard genetic algorithm is made better and it improved the system's performance by reducing the costs by 45% and execution time by 50% [21].

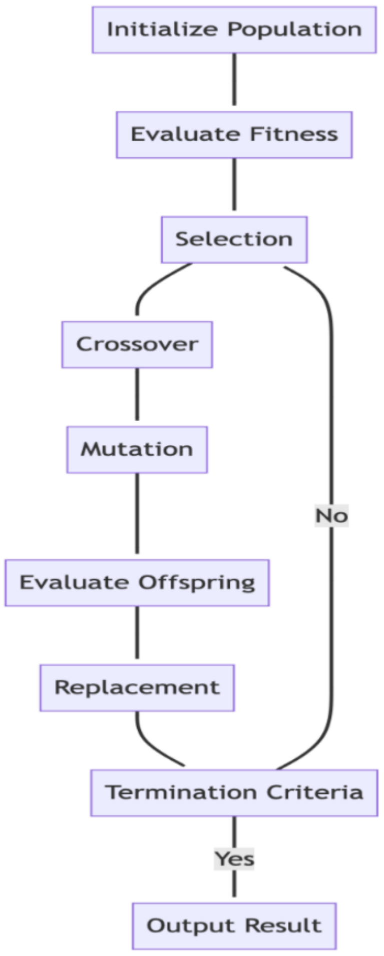

Fig. 3. Flowchart for hybrid job scheduling algorithm using genetic algorithm and fuzzy theory.

Algorithm

```
1. Initialize Population:
   - GenerateInitialPopulation()
2. Evaluate Fitness:
   - For each individual in population:
     - EvaluateFitness(individual)
3. Selection
 - Select individuals based on their fitness values
     - SelectionProcess()
4. Crossover:
   - For each pair of selected individuals:
     - PerformCrossover()
5. Mutation:
   - For each offspring:
     - ApplyMutation()
6. Evaluate Offspring:
   - For each offspring:
     - EvaluateFitness(offspring)
7. Replacement:
   - Replace individuals in the current population with offspring:
     - ReplacePopulation()
8. Termination Criteria:
   - Check if termination criteria are met:
     - If TerminationCriteriaMet():
       - TerminateAlgorithm()
     - Else:
       - Go to step 3
9. Output Result:
   - Output the best job assignment found in the population
     - OutputBestSolution()
```

3.2 | Round Robin Scheduling Algorithm

Another efficient load balancing algorithm is proposed which focuses on organizing VMs based on their importance and balancing the workload using a scheduling technique called round robin. VMs are sorted by comparing the key factors like bandwidth, RAM, and processing speed (MIPS). The algorithm employs round robin scheduling to evenly distribute tasks across VMs, ensuring that no single VM is overwhelmed or left idle for too long.

By doing this, the method prevents any VM from being overloaded, leading to better use of resources. CloudSim was used to evaluate the algorithms and the results shows that this method of task scheduling was faster than the traditional Round Robin Algorithm, indicating its efficiency in managing workloads in a cloud environment [22].

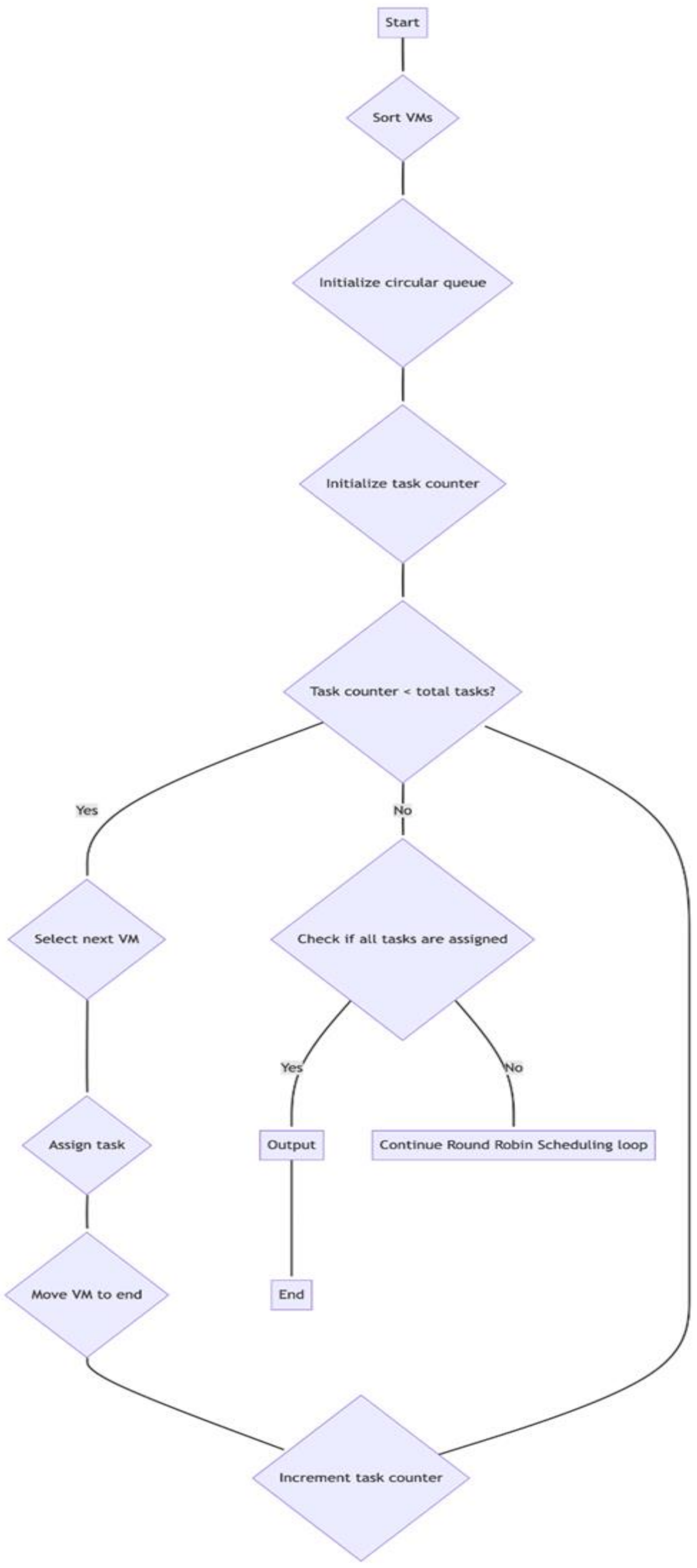
Start
Sort VMs
Initialize circular queue
Initialize task counter
Task counter < total tasks?
Yes
No
Select next VM
Check if all tasks are assigned
Yes
No
Assign task
Output
Continue Round Robin Scheduling loop
Move VM to end
End
Increment task counter

Fig. 4. Flowchart for load balancing using round robin scheduling.

Algorithm

```
1. Start:
   - Begin the algorithm.
2. Input Tasks and Resources:
   - Input the list of tasks to be scheduled and the available resources.
3. Initialize Variables:
   - Initialize variables for task scheduling status (e.g., task completion status).
   - Initialize variables for resource utilization tracking.
4. Apply LBMM Algorithm:
   - Use LBMM algorithm to initially allocate tasks to resources based on minimum execution time.
   - Update resource utilization accordingly.
5. Check Resource Utilization:
   - Evaluate the current resource utilization against a predefined threshold.
   - If utilization exceeds the threshold, proceed to step 6. Otherwise, skip to step 8.
6. Apply OLB Algorithm:
   - Identify underutilized resources using the OLB algorithm.
   - Reallocate tasks from highly utilized resources to underutilized ones to balance the load.
7. Update Resource Utilization:
   - Update resource utilization based on the task reallocation performed in step 6.
8. Check Task Completion:
   - Check if all tasks have been completed.
   - If all tasks are completed, proceed to step 10. Otherwise, continue to step 9.
9. Repeat LBMM and OLB:
   - If not, all tasks are completed, repeat steps 4 to 7 iteratively until all tasks are completed.
10. End:
    - Finish the algorithm.
```

3.3 | Opportunistic Load Balancing and Load Balancing Min-Min

A two-phase scheduling load balancing algorithm is proposed which combines two existing algorithms – Opportunistic Load Balancing (OLB) and Load Balancing Min-Min (LBMM) scheduling algorithms. The OLB scheduling algorithm makes sure that all the nodes in the system stay active, maintaining a balanced workload distribution, which aims to prevent any node from being overloaded or underutilized, enhancing overall system efficiency.

The LBMM algorithm aspect focuses on minimizing the time it takes to complete each task on a node, thus reducing the overall completion time of all tasks in the system. By optimizing task execution time, LBMM enhances utilization and work efficiency. By combining LBMM and OLB together, the algorithm benefits from both strategies: ensuring workload balance across nodes and minimizing task execution time. This combination enhances resource utilization and overall work efficiency in the system, contributing to better performance and faster task completion [22].

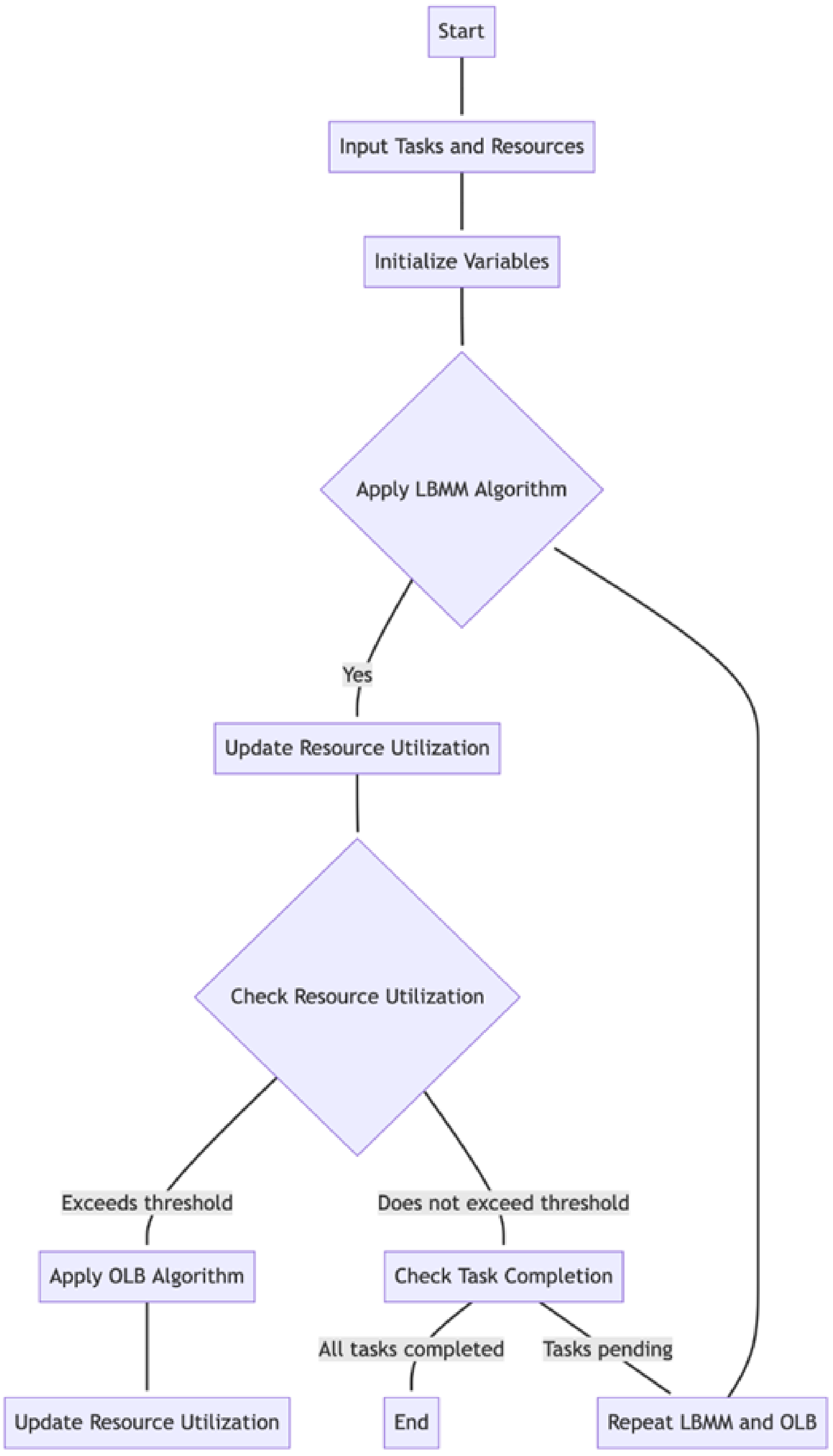

Fig. 5. Flowchart for two-phase scheduling algorithm – OLB + LMBB.

Algorithm

1. Start:

   - Begin the algorithm.

2. Input Tasks and Resources:

   - Input the list of tasks to be scheduled and the available resources.

3. Initialize Variables:

- Initialize variables for task scheduling status (e.g., task completion status).
- Initialize variables for resource utilization tracking.

4. Apply LBMM Algorithm:
   - Use LBMM algorithm to initially allocate tasks to resources based on minimum execution time.
   - Update resource utilization accordingly.
5. Check Resource Utilization:
   - Evaluate the current resource utilization against a predefined threshold.
   - If utilization exceeds the threshold, proceed to step 6. Otherwise, skip to step 8.
6. Apply OLB Algorithm:
   - Identify underutilized resources using the OLB algorithm.
   - Reallocate tasks from highly utilized resources to underutilized ones to balance the load.
7. Update Resource Utilization:
   - Update resource utilization based on the task reallocation performed in step 6.
8. Check Task Completion:
   - Check if all tasks have been completed.
   - If all tasks are completed, proceed to step 10. Otherwise, continue to step 9.
9. Repeat LBMM and OLB:
   - If not, all tasks are completed, repeat steps 4 to 7 iteratively until all tasks are completed.
10. End:
    - Finish the algorithm.

3.4 | Bee Life Algorithm and Greedy Approach

A modified task scheduling algorithm is proposed which combines the concepts of Bee Life Algorithm (BLA) and a greedy algorithm to optimize service in a hybrid cloud environment. BLA is primarily responsible for task scheduling in the hybrid cloud setup. It is a metaheuristic algorithm inspired by the foraging behaviour of honey bees. It copies the way bees explore their environment to find the best sources of nectar. Similarly, based on certain criteria, BLA would assign tasks to different resources in the cloud, aiming to optimize performance metrics such as makespan or resource utilization.

In addition to BLA, the algorithm makes locally optimal choices at each step with the hope of finding a global optimum. In that case, the greedy algorithm selects one data center randomly for task allocation. In this process, jobs are processed using a non-preemptive priority queue. This means that tasks are executed in the order they are received, and it cannot be interrupted until it completes [23].

This modified algorithm employs a hybrid approach where tasks first enter the BLA algorithm for initial scheduling. BLA optimizes the task allocation based on the heuristics, providing a good initial solution. However, BLA alone might not find the optimal solution due to its exploration-exploitation tradeoff. After BLA processing is done, the algorithm also utilizes a greedy method to select a data center randomly. By combining BLA and the greedy algorithm, the modified task scheduling algorithm aims to reduce the makespan, which is the total time taken to complete all tasks.

Overall, this hybrid approach leverages the strengths of both BLA and greedy algorithms to achieve an optimal or near-optimal task scheduling algorithm in a hybrid cloud environment [20].

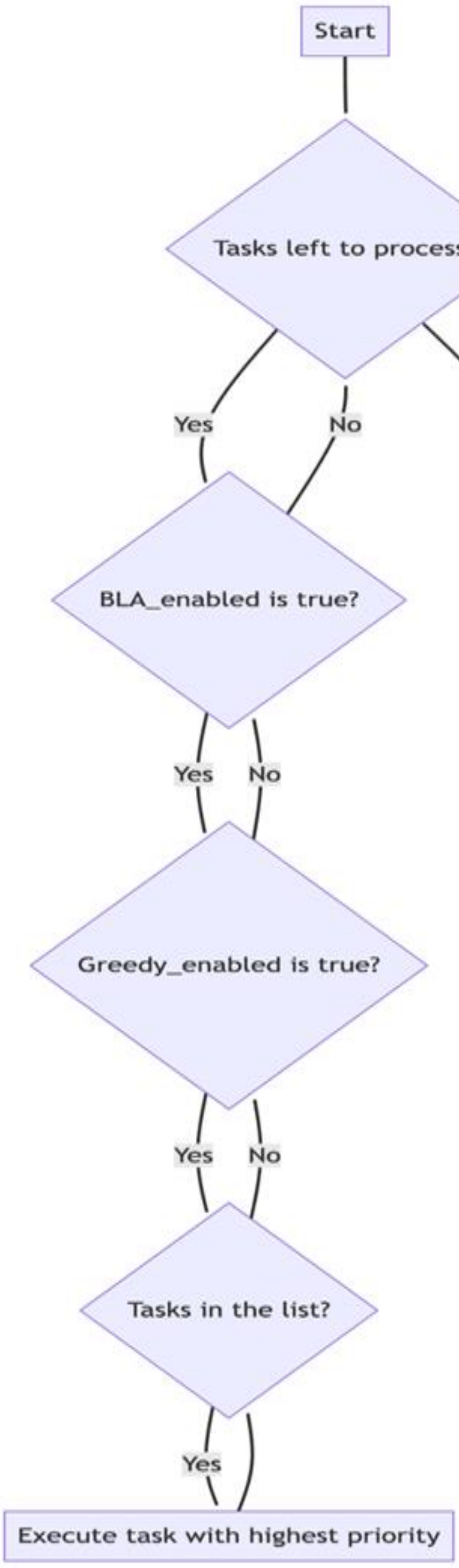

Fig. 6. Flowchart for BLA algorithm and greedy approach.

Algorithm

```
1. Start:
- Create a list to store tasks.
-Set BLA_enabled to true and Greedy_enabled to true.
2. Define a function ModifiedTaskScheduling( ):
-While there are tasks left to process
-Receive a new task and add it to the list.
3. If BLA_enabled is true:
-Use BLA to assign tasks to resources.
4. If Greedy_enabled is true:
-Use Greedy Algorithm to randomly select a data center.
5. Process tasks in the list:
6. While there are tasks in the list:
-Execute the task with the highest priority.
7. End Function
- End
```

4 | Analysis

In this section, we have analyzed the selected algorithms which aims to optimize task scheduling in hybrid cloud environments. Here are the results for each algorithm. The Genetic algorithm and Fuzzy Theory demonstrated significant improvements in system performance. It achieved a notable reduction in costs by 45% and execution time by 50%. This indicates enhanced resource utilization and cost-effectiveness in managing workloads in hybrid cloud environments. The Round Robin algorithm showed efficiency in workload management compared to the traditional Round Robin Algorithm. It effectively balanced workloads across VMs, leading to improved resource utilization and optimized performance in a cloud environment. By integrating OLB and LBMM, the algorithm enhanced resource utilization and overall work efficiency.

It achieved better performance and faster task completion by balancing workload distribution and minimizing task execution time across nodes in the system. By leveraging the strengths of both BLA and the greedy algorithm, the proposed approach aimed to achieve an optimal or near-optimal task scheduling algorithm. It focused on reducing the makespan, which is the total time taken to complete all tasks, leading to improved system performance and efficiency in hybrid cloud environments. In summary, these algorithms offer promising solutions to address load balancing challenges in hybrid cloud environments. They demonstrate improvements in system performance, efficiency, and resource utilization, contributing to better overall performance and management of workloads in hybrid cloud setups. Analysis of the studied algorithms shown in *Table. 1*.

Table. 1. Analysis of the studied algorithms.

Algorithm	Throughput	Latency	Performance	Execution Time
Genetic + Fuzzy Theory	High	Low	Significant	Reduced
Round Robin	Moderate	Moderate	Efficient	Moderate
OLB + LBMM	High	Low	Enhanced	Reduced
BLA + Greedy	High	Low	Optimal	Reduced

5 | Conclusion

In today's digital landscape, cloud computing has revolutionized how businesses manage their IT infrastructure. With the prevalence of hybrid clouds, which combine both private and public cloud resources, effective load balancing has become paramount. Load balancing ensures that tasks are evenly distributed across various cloud services, optimizing resource utilization and enhancing system efficiency. This research explores the challenges associated with load balancing in hybrid cloud environments and proposes strategies to overcome them, providing valuable insights for organizations navigating this dynamic landscape.

Hybrid clouds have become increasingly common among enterprises, offering a flexible and scalable solution to meet diverse IT needs. However, managing workloads across hybrid cloud infrastructures presents unique challenges. One such challenge is ensuring efficient load balancing, which involves distributing tasks across cloud services in a way that maximizes performance and minimizes resource wastage. To address these challenges, various load balancing algorithms have been proposed and evaluated. In conclusion, managing workloads in hybrid cloud environments is crucial for businesses using both private and public cloud resources. By using these strategies, companies can save money, work more efficiently, and keep their customers happy.

Author Contributions

Akash Kumar contributed to conceptualization, methodology, and drafting the manuscript. Mohit Tiwary handled formal analysis, data curation, and contributed to writing – review & editing. Ritwik Dev Singh was involved in the investigation and visualization, and provided supervision.

Funding

This research received no external funding.

Data Availability

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] Buyya, R., Broberg, J., & Goscinski, A. M. (2010). *Cloud computing: principles and paradigms*. John Wiley & Sons.

[2] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A. (2010). A view of cloud computing. *Communications of the acm*, *53*(4), 50–58. https://dl.acm.org/doi/fullHtml/10.1145/1721654.1721672

[3] Mohapatra, H., Kolhar, M., & Dalai, A. K. (2024). Efficient energy management by using sjf scheduling in wireless sensor network. *International conference on advances in distributed computing and machine learning* (pp. 211–221). Springer. https://link.springer.com/chapter/10.1007/978-981-97-1841-2_15

[4] Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, *1*(1), 7–18. https://doi.org/10.1007/s13174-010-0007-6

[5] Balaji, K. (2021). Load balancing in cloud computing: issues and challenges. *Turkish journal of computer and mathematics education (turcomat)*, *12*(2), 3077–3084.

[6] Katyal, M., & Mishra, A. (2014). *A comparative study of load balancing algorithms in cloud computing environment*. ArXiv Preprint ArXiv:1403.6918.

[7] Mohapatra, H. (2021). Socio-technical challenges in the implementation of smart city. *2021 international conference on innovation and intelligence for informatics, computing, and technologies (3ICT)* (pp. 57–62). IEEE. DOI: 10.1109/3ICT53449.2021.9581905

[8] Marinescu, D. C., Paya, A., Morrison, J. P., & Olariu, S. (2017). An approach for scaling cloud resource management. *Cluster computing*, *20*, 909–924.

[9] Arunarani, A. R., Manjula, D., & Sugumaran, V. (2019). Task scheduling techniques in cloud computing: a literature survey. *Future generation computer systems*, *91*, 407–415. https://www.sciencedirect.com/science/article/pii/S0167739X17321519

[10] Xin, Y., Xie, Z.-Q., & Yang, J. (2017). A load balance oriented cost efficient scheduling method for parallel tasks. *Journal of network and computer applications*, *81*, 37–46. https://www.sciencedirect.com/science/article/pii/S1084804516303496

[11] Mesbahi, M., & Rahmani, A. M. (2016). Load balancing in cloud computing: a state of the art survey. *International journal modern education and computer science*, *8*(3), 64.

[12] Mishra, S. K., Sahoo, B., & Parida, P. P. (2020). Load balancing in cloud computing: A big picture. *Journal of king saud university - computer and information sciences*, *32*(2), 149–158. https://www.sciencedirect.com/science/article/pii/S1319157817303361

[13] Parida, B. R., Rath, A. K., Pati, B., Panigrahi, C. R., Mohapatra, H., & Buyya, R. (2024). SSEPC cloud: carbon footprint aware power efficient virtual machine placement in cloud milieu. *Computer science and information systems*, 18.

[14] Radha, K., Rao, B., Babu, S., Rao, K., Reddy, V., & Saikiran, P. (2014). Allocation of resources and scheduling in cloud computing with cloud migration. *International journal of applied engineering research*, *9*(19), 5827–5837.

[15] Mishra, S., Sahoo, B., & Parida, P. (2018). Load balancing in cloud computing: a big picture. *Journal of king saud university*, *32*. DOI:10.1016/j.jksuci.2018.01.003

[16] Gunasekhar, T., Rao, K. T., Reddy, V. K., & Rao, B. T. (2015). Mitigation of insider attacks through multi-cloud. *International journal of electrical & computer engineering (2088-8708)*, *5*(1).

[17] Mukherjee, S., & Mohapatra, H. (2024). Performance analysis of different manet routing protocols. *2024 5th international conference on innovative trends in information technology (ICITIIT)* (pp. 1–6). IEEE.

[18] Parsa, S., & Entezari-Maleki, R. (2009). RASA: a new grid task scheduling algorithm. *International journal of digital content technology and its applications*, *3*(4), 91–99.

[19] Milani, A. S., & Navimipour, N. J. (2016). Load balancing mechanisms and techniques in the cloud environments: systematic literature review and future trends. *Journal of network and computer applications*, *71*, 86–98. https://www.sciencedirect.com/science/article/pii/S1084804516301217

[20] Javanmardi, S., Shojafar, M., Amendola, D., Cordeschi, N., Liu, H., & Abraham, A. (2014). Hybrid job scheduling algorithm for cloud computing environment. *Proceedings of the fifth international conference on innovations in bio-inspired computing and applications ibica 2014* (pp. 43–52). Springer.

[21] Bhowmik, S., Biswas, S., Vishwakarma, K., & Chattoraj, S. (2016). An efficient load balancing approach in a cloud computing platform. *Journal of computer engineering (IOSR-JCE)*, *18*(6), 85–89. https://www.academia.edu/download/77764880/O1806068589.pdf

[22] Wang, S. C., Yan, K. Q., Wang, S. S., & Chen, C. W. (2011). A three-phases scheduling in a hierarchical cloud computing network. *2011 third international conference on communications and mobile computing* (pp. 114–117). IEEE. DOI: 10.1109/CMC.2011.28

[23] Mizan, T., Masud, S., & Latip, R. (2012). Modified bees life algorithm for job scheduling in hybrid cloud. *International journal of engineering and technology*, *2*(6), 974–979.

[24] Mohapatra, H., & Mishra, S. R. (2024). Unlocking insights: exploring data analytics and ai tool performance across industries. In *Data analytics and machine learning: navigating the big data landscape* (pp. 265–288). Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-97-0448-4_13