Hindawi
Wireless Communications and Mobile Computing
Volume 2024, Article ID 2357142, 9 pages
https://doi.org/10.1155/2024/2357142

Research Article

Hybrid Agent-Based Load-Balancing Approach Used in an IaaS Platform

Shoney Sebastian,[1] **Iyyappan M.**,[2] **Sultan Ahmad**,[3,4] **Mohammad Maqbool Waris**,[5] **Hikmat A. M. Abdeljaber**,[6] **and Jabeen Nazeer**[3]

[1]*Department of Computer Science, CHRIST University, Hosur Road, Karnataka, Bengaluru 560029, India*
[2]*College of Information Technology, Ahlia University, P.O. Box 10878, Manama, Bahrain*
[3]*Department of Computer Science, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, P.O. Box 151, Alkharj 11942, Saudi Arabia*
[4]*University Center for Research and Development (UCRD), Department of Computer Science and Engineering, Chandigarh University, Gharuan, Mohali 140413, Punjab, India*
[5]*Department of Mechanical Engineering, Adama Science and Technology University, Adama, Ethiopia*
[6]*Department of Computer Science, Faculty of Information Technology, Applied Science Private University, Amman, Jordan*

Correspondence should be addressed to Mohammad Maqbool Waris; mohammad.waris@astu.edu.et

Received 27 February 2023; Revised 30 March 2024; Accepted 15 April 2024; Published 26 April 2024

Academic Editor: Qiang Ye

Cloud computing has received a resounding welcome. It was created following methodical and thorough study in web services, distributed computing, utility computing, and virtualization, and it offers several benefits, including lower costs, less space required, and easier management. These advantages bring in a significant number of new users to the cloud platform every day. In addition, because cloud computing is an Internet-based computing paradigm, it must deal with the issue of overwhelming demands through effective load-balancing. A very small number of studies only focus on load-balancing problems in cloud computing platforms, while the majority of load-balancing research is accessible in many domains, including parallel, distributed, and grid computing. Infrastructure as a Service (IaaS), Software as a Service, and Platform as a Service are the three basic categories under which cloud computing falls. For these models, there are notable differences in the load-balancing techniques used. This work compared the outcome with the current method and presented a hybrid agent-based load-balancing approach for the IaaS platform.

1. Introduction

In the distributed computing paradigm, Armbrust et al. [1] and Uddin and Ahmad [2] explained that cloud computing provides an environment to access virtualized hardware and software over the internet. Grossman [3] mentioned that virtualization is one of the essential characteristics of cloud computing. Lee and Zomaya [4] and Aliabadi and Ahmadi [5] provide a feasible solution for managing physical resources dynamically. It brings out a significant reduction in the number of physical servers required in the data center, as a result, increases resource utilization and decreases the management complexity. Sotomayor et al. [6], through virtualization technologies, mapping multiple virtual machines (VMs) on physical servers ensures the entire system's resource utilization dynamically. However, since the resources in cloud computing platforms are highly vibrant and heterogeneous, VMs need to be dynamically adaptive to the cloud platform to achieve maximum performance. For that, the available physical resources must be efficiently allocated, and a proper load-balancing strategy needs to be implemented. Since the arrival of jobs in cloud computing is not predictable and individual nodes connected to the cloud vary in their physical capacities, many crucial problems such as performance, security, and availability need to be addressed for the realization of cloud computing. Efficient utilization of the resources and a proper sharing of load among available resources increase the cloud-based application's performance. Load balancer plays a significant role in this regard. Azeez [7] load-balancing strategy can be static, dynamic, or mixed. Amazon EC2 replicates

instances of the middleware platform for web services to achieve dynamic load balancing. When the load exceeds the predefined threshold, new instances of the platform are started [7].

This paper proceeds with the details of related works in Section 2, a review of the literature, the proposed method in Section 3, Section 4 proposed algorithm, Section 5 experiment and result analysis, and Section 6 conclusion.

2. Literature Review

As per Chow and Kohler [8] and Ni and Hwang [9], to balance the resources of the computing environment the loads are equally distributed into the parallel executions. In traditional computing environments, such as distributed, parallel, and grid computing, several static, dynamic, and mixed scheduling strategies, are proposed by the researchers. Load-balancing techniques can be centralized or distributed. In the centralized approach [8, 9], the load-balancing decision is taken in one place and communicated to the entire system. This strategy ensures better load-balancing decisions at the cost of reduced scalability. Corradi et al. [10] proposed the diffusive policy for the equal distribution load but to accomplish the task rebalancing aim, volatile concurrent programs must be executed efficiently. This needs the running distribution choices. Designing a universal allotment mechanism is challenging, as it must react with the fluid nature of the application being used [10]. A diffusive policy can accomplish universal leveling in fixed-load scenarios by combining separate operations. This is caused by the entire penetration across all zones. The preceding summary of diffusive rules does not include regulations that need worldwide alignment of their behavior. In a comparable manner it eliminates strategies that employ dispersion to spread workload data as well as provide a global view of the system. None of these methods is expandable, because they are challenging to set up in substantially comparable infrastructures. The dispersion strategy may act as a basis for a variety of approaches. The way they are implemented ought to tackle a number of challenges during the neighborhood decision-making stage, which needs to occur preceding behavior or load migrations. Specifically, for every node: The beginning stage was the selection element, which sets the circumstances that initiate the subsequent rebalancing stages. The situation assessment stage determines if networks across the domain are going to require overload measures. The placement element is an assessment part that detects the region's underloaded vertices (users) and overloaded vertices (transmitters) that require intervention. The choice stage is an assessment part that determines what data to transfer from source to destination networks.

Kale [11], meanwhile, distributed approaches are designed to be scalable but yield poor quality load distribution on extremely large machines and take more time to give good solutions in a very dynamic environment. Adaptive balanced load solutions must be used for concurrent execution of unexpectedly designed calculations. I evaluated the efficacy with two such techniques. The acquiring inside area tackle involves sending fresh targets down the greatest demand grade to the regional optimum inside a certain distance of the origin of simultaneous execution. The objective cannot be returned by this execution later. The gradient mechanism is greater in sophistication. The system aims to shift labor between individuals with ample parallel to people who are at risk of becoming unproductive. As standard, the code is stored on-site and transmitted forth after an unused router is detected. It requires an unfinished entity to fulfill the balance of load duties. One of the main problems of the existing paper cloud services in terms of security and usability has been addressed by global communication. The study supports cloud federation as a solution to these kinds of cloud service users' needs. By decreasing execution times and the number of users who had to wait, as well as by improving the status of user requests, the cloud federation suggested in this study has addressed the problems previously discussed. By calculating on progress tasks corresponding to the number of VMs, as detailed in the existing technique, the suggested system has demonstrated that the central management system can detect the idle or less loaded VM. Using subscription identities to access cloud resources from several cloud service providers serves as an example. The time consumption is effectively reduced when CSCs employ multiple identities to access resources on several CSPs, as the results clearly show. The execution time is shortened since the subscription ID is sufficient to access the service without the need for federated clouds, based on all the algorithms and results received; this lessens the workload on the central management system that 3,115 ms are needed to complete 25 LogIn operations. Zheng et al. [12] and Jha et al. [13] discussed about the hybrid approach in the paper, which overcomes the drawbacks of both centralized and distributed approaches.

Sotomayor et al. [6] discussed a well-known algorithm known as round robin (RR) with the least connection. In this technique, the request is forwarded to the node having the least number of connections. At some point in time, some nodes may be heavily loaded, and others remain idle. Radojević and Žagar [14] suggested an improved version of RR known as the Central Load-Balancing Decision Model. In this technique, the connection time between the client and the node is calculated, and if it exceeds the threshold, it disconnects the connection and forwards the request to a further node using the RR approach.

Blum [15] discussed about a load-balancing technique called ant colony optimization. In this approach, once the request is generated, the ant and the pheromone move in the forward direction from an overloaded node in search of an underloaded node. After locating an underloaded node, it continues its search further to the next overloaded node and it backtracks to the last underloaded node. The research paper of Wang et al. [16] suggests a dynamic load-balancing algorithm called load-balancing min–min (LBMM) based on a three-level hierarchy. LBMM uses an opportunistic load-balancing algorithm, which ensures each node is busy in the cloud without considering the execution time of the node, which causes a bottleneck in the system. Singh et al. [17] In the autonomous agent-based load-balancing algorithm (A2LB) for load redistribution in an overloaded VM [17].

Some parameters are observed here as follows:

(i) The data are segregated into the cloud.
(ii) The machines are virtually connected to data.
(iii) Fitness methodologies are used to load the data.
(iv) The request is used to fetch the data from the cloud.
(v) The load will be allocated from the table of a VM. If it is overloaded with data, the VM investigates an alternate process of cloud segregation.
(vi) Once this process is completed, the passage agent sent an acknowledgment to the allocation table.

These are the few observations in the existing algorithm as follows:

(1) A cloud architecture by Jinhua et al. [18] shows the presence of multiple data centers with thousands of nodes inside them. Each node will have multiple numbers of VMs [18]. Cherkasova et al. [19] stated that since each VM information is stored in the virtual_machine_load fitness table in the A2LB approach, the size of the table will be extremely large. So, the load agent will take more time to scan this table [19].
(2) No policy was mentioned for diverting the load request coming into the data center. Since a cloud provider will have multiple data centers, the policy used for sending the load request to different data centers is not mentioned. The cloud Infrastructure designed by a cloud provider consists of multiple clusters and multiple zones within a cluster spread across the world. The log file of VMs maintained in the cloud infrastructure bottleneck the performance of the cloud systems as the log fie size increases.
(3) Sotomayor et al. [20] stated that this algorithm checks the status of a VM and, if the VM has enough space to accommodate the incoming request, passes the request to the VM. But in the actual scenario, especially in the IaaS platform, instead of checking the status of the VM, it is better to check the status of the node and, if the node is free or in the normal state, pass the request to the node.
(4) Alqahtani et al. [21] stated that the load-balancing algorithm focuses on the time-based resource allocation of nodes, which can effectively perform the operation and execution in a timely manner. This paper focuses on the failure rate calculation, but this module is required to support the higher level of the virtualization process. Calculating the failure rate is a difficult process for the virtualization of software allocation [21].
(5) Nasr et al. [22] stated that cloud let simulating algorithm followed the two different strategies for the virtualization of the scheduling algorithm. In this paper, more focus is on running time and reduce the complexity of using the hybrid approach [22]. Our proposed model utilized our time-consuming process and increased the resource utilization.

3. Proposed Method

Cloud provides three basic services named Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). In IaaS, the resources such as CPU, memory, disk space, etc. can be availed to the requesting client in the form of a service. Cherkasova et al. [19] proposed the A2LB algorithm that was redesigned to work effectively in the IaaS platform. The hybrid model has become a prominent player in the constantly changing field of cloud computing, providing enterprises with the greatest features of both on-premises infrastructure—security and control—and public cloud computing—scalability and agility. Getting the proper balance: The demands and priorities of businesses vary. Private clouds may be necessary for sensitive data due to their strict control, while public clouds are ideal for workloads that fluctuate and offer more flexibility. You may customize your cloud infrastructure using a hybrid to meet certain requirements. While new projects take use of the flexibility of the public cloud, legacy applications might remain on-premises for security reasons. This equilibrium maximizes cost-effectiveness, security, and performance. Scalability on demand: Rapid scaling is a strength of public clouds, making them ideal for workload fluctuations. There is a price for this freedom, though. By using a hybrid strategy, you may address demand surges by scaling up and down in the public cloud during peak hours without going over budget for permanent resources. You may retain cost efficiency by reducing the amount of your on-premises infrastructure after things settle down. Data residency and compliance: Data must comply with certain standards and remain inside certain geographic limits. Utilizing the public cloud for nonessential functions while maintaining sensitive data on-premises for compliance is possible with a hybrid cloud. This maintains regulatory compliance while maintaining cloud computing's advantages. Preventing vendor lock-in: Although public cloud providers provide appealing benefits, depending too much on one might restrict your freedom and alternatives [23, 24]. You remain open with hybrid cloud. For certain purposes, you may choose a variety of public cloud providers, preventing lock-in and guaranteeing that you receive the finest services available for each activity. Encouraging Innovation: The hybrid paradigm dismantles boundaries between public cloud and on-premises systems [25]. Collaboration and creativity are facilitated by connecting data and applications between the two. Resources are readily available to developers, encouraging more rapid experimentation and agile development. Businesses may create a custom cloud architecture that optimizes security, agility, and compliance with the help of the hybrid approach, which goes beyond just balancing prices and performance. It has nothing to do with staying off-site or avoiding public clouds. It is important to select the appropriate tool for the job, and the hybrid approach gives you the freedom to do so. Recall that there is no one-size-fits-all explanation for the significance of the hybrid approach. Every firm has certain requirements and limitations. But, by being aware of the hybrid model's main advantages, you may decide on your cloud approach and take use of its advantages to meet your

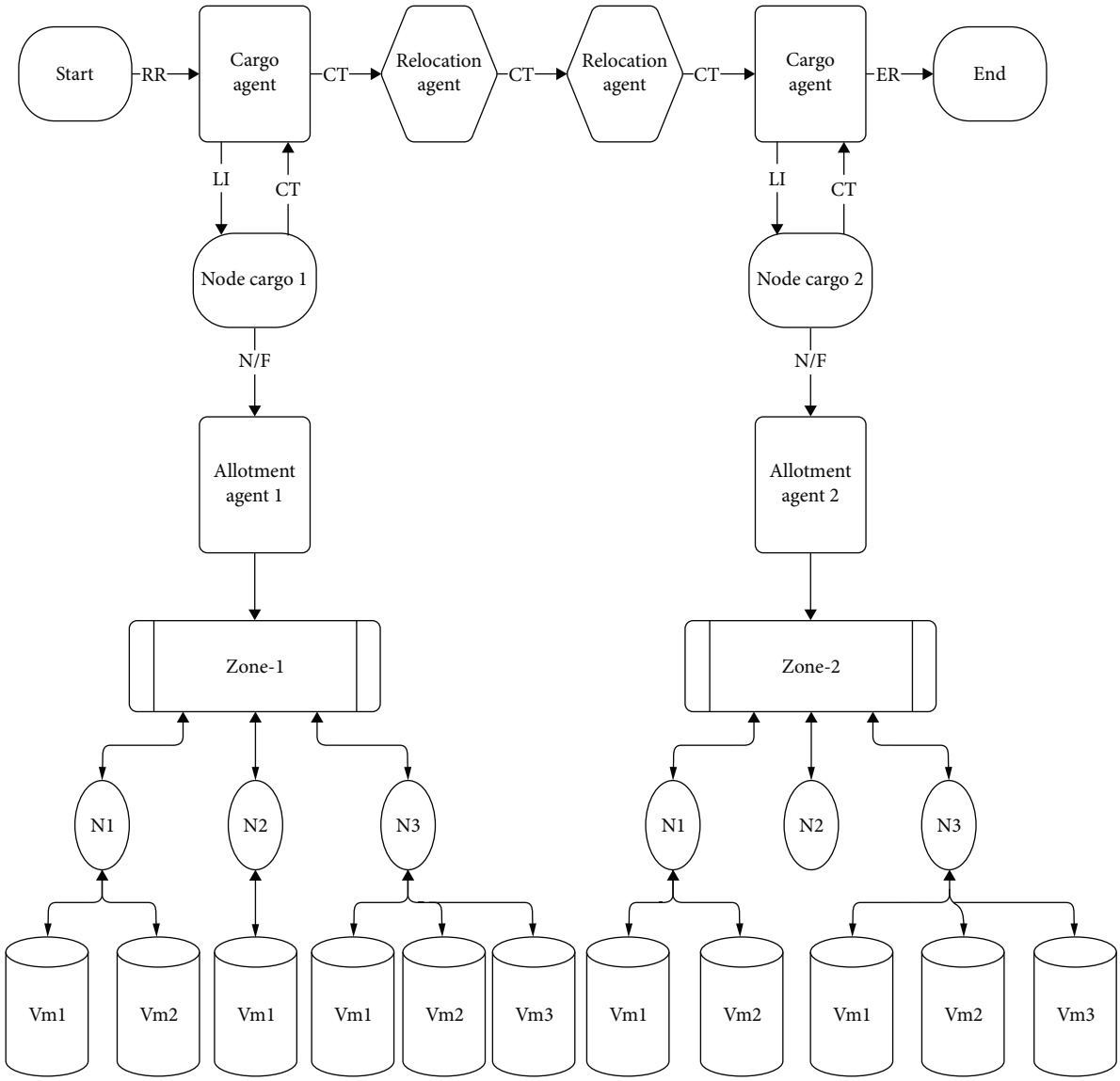

Figure 1: Flow of process for multiple data centers.

Table 1: Zone resource table.

Z_Id	Nd_Id	Π	μ	Ω

Table 2: Zone utilization table.

Z_Id	Nd_Id	α	β	∞	V	S

Table 3: VM cargo table.

Z_Id	Nd_Id	V_Id

company's objectives. The cloud provider depicted in Figure 1 may have several data centers spread across several geographic regions, each referred to as a zone. A zone consists of several nodes gathered. IaaS creates virtual instances in nodes and sends them to clients based on requests from clients. Multiple virtual instances can exist within a single node. A single Amazon EC2 t2.nano instance, for instance, offers 0.5 GiB of RAM, one virtual CPU, and secondary storage using Amazon EBS.

A hybrid approach for load balancing is applied in the proposed work. Table 1 shows the structure of a zone resource. Z_Id stands for zone id, Nd_Id stands for node id, π stands for total CPU capacity, μ stands for total memory capacity, and Ω for the total storage capacity of a node in each zone. Table 2 describes the zone-level resource utilization. α indicates the amount of CPU used in a node, β stands for storage used in a node, stands for memory used in a node, v represents the fitness value of a node, and s represents the load status of the node. Table 3 keeps the records of the VMs available in a node of a specific zone. Tables 1–3 are stored in a distributive manner in each zone.

```
class VirtualMachine:
    def __init__(self, vm_id, capacity):
        self.vm_id = vm_id
        self.capacity = capacity
        self.current_workload = 0
    def assign_task(self, task):
        self.current_workload += task
    def get_utilization(self):
        return self.current_workload/self.capacity
class LoadBalancer:
    def __init__(self, vms):
        self.vms = vms
    def balance_load(self, task):
        # Simple load balancing: Assign the task to the VM with the lowest current workload
        target_vm = min(self.vms, key = lambda vm: vm.current_workload)
        target_vm.assign_task (task)
    def print_utilization(self):
        for vm in self.vms:
            print (f"VM {vm.vm_id} Utilization: {vm.get_utilization()}")
# Example usage
if __name__ == "__main__":
    # Create virtual machines
    vm1 = VirtualMachine (vm_id = 1, capacity = 120)
    vm2 = VirtualMachine (vm_id = 2, capacity = 170)
    vm3 = VirtualMachine (vm_id = 3, capacity = 130)
    vms = [vm1, vm2, vm3]
    # Create a load balancer
    load_balancer = LoadBalancer (vms)
    # Simulate tasks and load balancing
    tasks = [30, 40, 15, 25, 10]
    for task in tasks:
        load_balancer.balance_load (task)
    # Print VM utilizations
    load_balancer.print_utilization()
```

Level 1 of Cargo process

Select: Accept the user's request.

Results: Distribute resources using a sophisticated algorithm

Case1:

Call Allotment_agent (Cargo info) and Master_Commit_Agent (Cargo info) for each node in Table 4 with status = "Empty."

Case 2:

For every node N_Node with Status = "Normal" in Table 2,

{

If (N_Node.Availablememory-requested_memory >25%)~{ Call Allotment_agent() Call Master_Commit_Agent() }

Case 3:

If (no node with the statuses "Free" or "Normal" is accessible in Table 2)

Call Passage_Agent();

Level 2 Allotment Process

(i) Update Tables 2 and 3 and create a VM in the chosen node with the needed cargo information.

(ii) Using the information in the Node Resource Table, compute the modified node's fitness value.

μtotal = μavailable [from Node_Resource_Table]–μused [According to Table 2]

v% = μavailable/μtotal × 100

if (v >75%)

```
    {
    Allotment_status = Empty
    }
    Otherwise, if (Allotment_status >25%)
    {
    Allotment_status = Normal
    }
    Otherwise, Allotment_status = critical
    Refresh Tables 2 and 3.
Agent for travel ()
{
    Provide: Get the container agency
    To obtain the Non-Critical DC, initiate the Relocation agent.
    Please forward the request to DC.
}
Relocation process()
{
    Scan Table 4
    Select the DC that is least loaded and give the Passage agent back the DCID.
} //sample can be shown in Figure 5.
Cargo Information: Master_Commit Agent
{
    Add the information of the cargo to Table 4.
    Determine the Zone's fitness value by using the information shown in Table outlined below:
    From the Prime DC Resource Table, μavailable = μtotal. Used from the Prime DC Cargo Table
    v% is equal to available divided by total times 100.
    The allocation status is empty if (v >75%)
    If (Allotment_status >25%), then } Else
    The allocation status is Normal.
    Otherwise, allocation_status = critical
    Make Table 4 updates
```

Algorithm 1: Algorithm for VM.

This algorithm followed certain procedures to execute the process of load-balancing and a hybrid-based approach, sharing the resource information with the help of virtualization to store the data. This initial process started with the request received (RR) from the client, which has been taken into consideration of cargo agent 1 and moved forward to the node cargo 1 module. The node cargo module depends on the look-into (LI) process, which has been redirected into allotment agent 1; this agent receives input through the normal process (N). Those agents depend upon the zone-level segregation to store the resource data with the help of multiple virtualizations. For executing this process, the connecting interface of nodes is required to direct into the VM. In this execution three different node (N1, N2, N3) utilized for VM (Vm1, Vm2, Vm3). Once this storage process is completed or full of required space, it will be redirected into the parallel components of Zone-2. The Parallel activities looking for the Relocation agent with the help of critical thinking (CT), which will be working on the concepts of reallocation of resource storage. Once the request is completed on the virtual storage, it will share the response as well as if we want to delete the stored information with the help of free execution (F). From the above diagram Figure 1 considers into parallel agents of Zone 1 and Zone 2 for storing and retrieving the component of the hybrid approach with the help of load-balancing execution. The final module of end request (ER) will terminate the process agent request-based response.

Table 4 stores the fitness value, load status, and resource consumption statistics for each zone, including CPU, memory, and storage. Table 5 contains the resource data, including total CPU, total RAM, and total disk storage for each zone. Tables 4 and 5 are kept up to date in a single area within the cloud provider space.

4. Proposed Algorithm

Five functions make up the suggested algorithm (Algorithm 1) Cargo Agent, Allotment Agent, Passage Agent, Master Commit Agent, and Relocation Agent. In every zone, Cargo Agent receives the request from the client. When the Cargo Agent has reviewed the zone's condition, he or she can call the

Table 4: Node cargo table.

Z_Id	CPU_Used	Memory used (μ)	Disk used	Fitness value (ν)	Status

Table 5: Primary DC resource table.

Z_Id	Tot_CPU	Tot_Mem	Tot_Disk

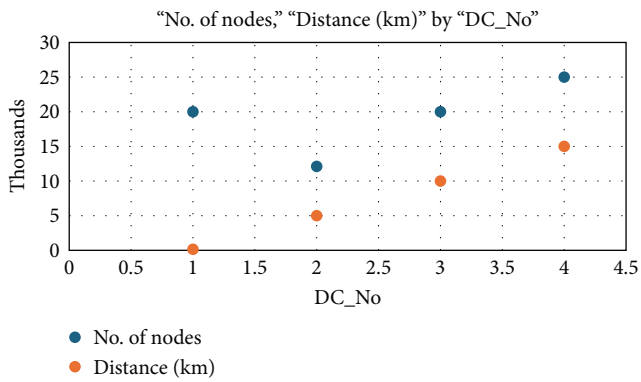

Figure 2: Distance measurements using nodes.

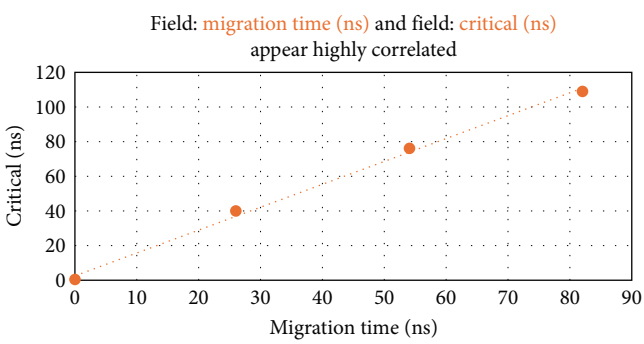

Figure 3: Cargo process of migration time.

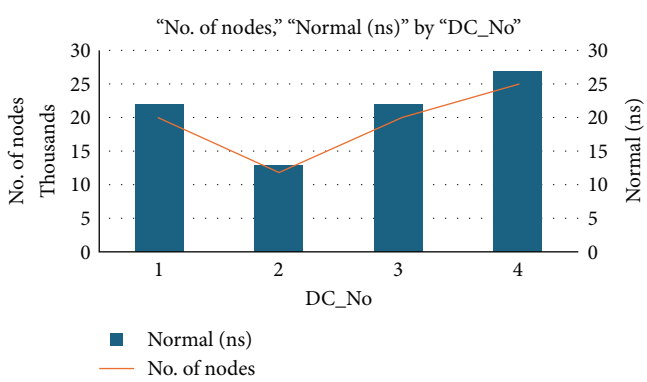

Figure 4: Allotment process of normal time.

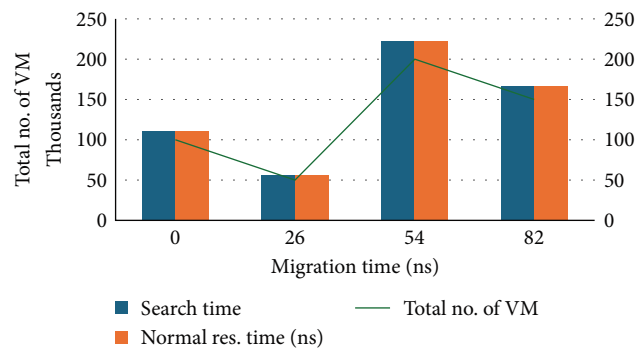

Figure 5: Search time for relocation process.

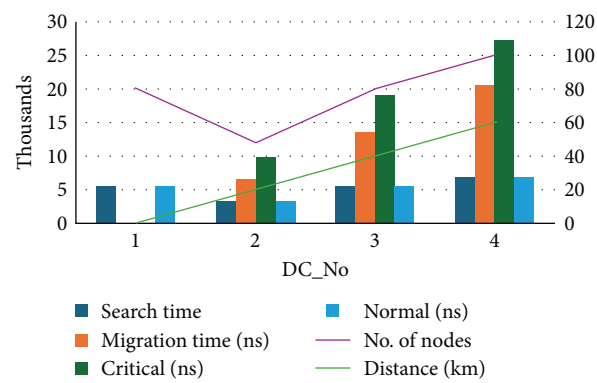

Figure 6: Performance analysis of the algorithm.

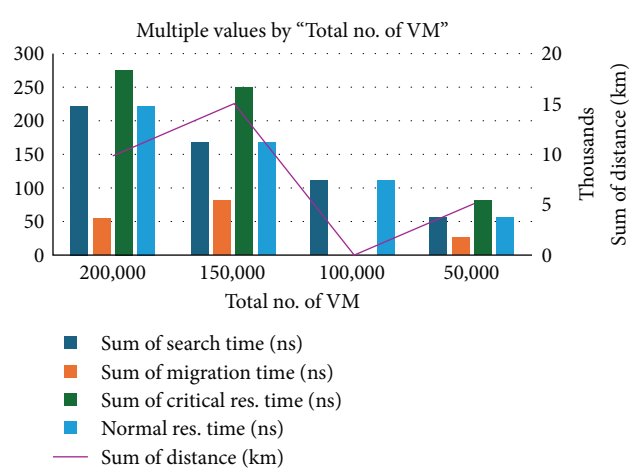

Figure 7: Comparison of various approaches.

Allotment Agent to assign the request to the Cargo Agent in that zone or the Relocation Agents to transmit it to the Cargo Agents in other zones. The mathematical problem illustrating the load-balancing usage processes is displayed below, and their result illustration can be shown in Figures 2–7.

The output of the suggested algorithm is displayed in Table 6. The calculation of response time under normal and critical situations involves assuming that there are four data centers, each with a distinct number of nodes. The

Table 6: Recommended algorithm.

DC_No	No. of nodes	Search time	Distance (km)	Migration time (ns)	Critical (ns)	Normal (ns)
1	20,000	22	0	0	0	22
2	12,000	13	5,000	26	39	13
3	20,000	22	10,000	54	76	22
4	25,000	27	15,000	82	109	27

Table 7: Compared algorithm.

DC_No	Total no. of VM	Search time (ns)	Distance (km)	Migration time (ns)	Critical res. time (ns)	Normal res. time (ns)
1	100,000	111.11	0	0	0	111.11
2	50,000	55.5	5,000	26	81.5	55.5
3	200,000	222.22	10,000	54	276.2	222.22
4	150,000	167	15,000	82	249	167

outcome of the existing method, which takes VM count into account rather than node count, is displayed in Table 7. A notable improvement in reaction time was seen in the new strategy, as demonstrated by the graphs in Figures 2 and 3. These figures were created using Tables 6 and 7.

5. Conclusion

The focus of the paper was to identify the challenges in the existing algorithm A2LB for the load distribution in the IaaS platform and refine it further to improve the performance. The proposed A2LB algorithm provides a better-clustered approach to resource utilization among user nodes and allocated memory space with an accurate load value. The virtual components also adhered to the various criteria for node point connection between the primary and secondary agents. It improves the optimization of CPU and memory usage while also reducing disc resource utilization. The scope of this work may be expanded in the future to include the metric parameter calculation of load-balancing in the SaaS platform.

Data Availability

The data used to support the findings of this study are available from the first author upon request (shoneyks@gmail.com).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

We thank the Deanship of Scientific Research, Prince Sattam Bin Abdulaziz University, Alkharj, Saudi Arabia, for help and support. This study is supported via funding from Prince Sattam Bin Abdulaziz University project number (PSAU/2024/R/1445).

References

[1] M. Armbrust, A. Fox, R. Griffith et al., "Above the clouds: a berkeley view of cloud computing," University of California at Berkeley Technical, Tech. Rep. No. UCB/EECS-2009-28, 2009.

[2] M. Y. Uddin and S. Ahmad, "A review on edge to cloud: paradigm shift from large data centers to small centers of data everywhere," in *2020 International Conference on Inventive Computation Technologies (ICICT)*, pp. 318–322, IEEE, Coimbatore, India, February 2020.

[3] R. L. Grossman, "The case for cloud computing," *IT Professional*, vol. 11, no. 2, pp. 23–27, 2009.

[4] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomputing*, vol. 60, pp. 268–280, 2012.

[5] M. R. Aliabadi and M. R. Ahmadi, "Proposing a comprehensive storage virtualization architecture with related verification for data center application," *Advanced in Information Sciences and Service Sciences*, vol. 2, no. 3, pp. 68–75, 2010.

[6] B. Sotomayor, K. Keahey, I. Foster, and T. Freeman, "Enabling cost-effective resource leases with virtual machines," in *In Hot Topics Session in ACM/IEEE International Symposium on High Performance Distributed Computing*, (HPDC 2007), 2007.

[7] A. Azeez, *Auto-Scaling Axis2 Web Services on Amazon EC2*, Apachecon Europe, Amsterdam, 2009.

[8] Y.-C. Chow and Kohler, "Models for dynamic load balancing in a heterogeneous multiple processor system," *IEEE Transactions on Computers*, vol. C-28, no. 5, pp. 354–361, 1979.

[9] L. M. Ni and K. Hwang, "Optimal load balancing in a multiple processor system with many job classes," *IEEE Transactions on Software Engineering*, vol. SE-11, no. 5, pp. 491–496, 1985.

[10] A. Corradi, L. Leonardi, and F. Zambonelli, "Diffusive load-balancing policies for dynamic applications," *IEEE Concurrency*, vol. 7, no. 1, pp. 22–31, 1999.

[11] L. V. Kale, "Comparing the performance of two dynamic load distribution," *Proceedings of the International Conference on Parallel Processing*, vol. 1, pp. 8–12, 1988.

[12] G. Zheng, A. Bhatelé, E. Meneses, and L. V. Kalé, "Hierarchical load balancing for large scale supercomputers," in *Third International Workshop on Parallel Programming Models and Systems Software for High-End Computing (P2S2)*, pp. 436–444, IEEE, San Diego, CA, USA, September 2010.

[13] S. Jha, A. Sultan, M. Alharbi, B. Alouffi, and S. Sebastian, "Secured and provisioned access authentication using subscribed user identity in federated clouds," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 11, 2021.

[14] B. Radojević and M. Žagar, "Analysis of issues with load balancing algorithms in hosted (cloud) environments," in *2011*

Proceedings of the 34th International Convention MIPRO, pp. 416–420, IEEE, Opatija, Croatia, May 2011.

[15] C. Blum, "Ant colony optimization: introduction and recent trends," *Physics of Life Reviews*, vol. 2, no. 4, pp. 353–373, 2005.

[16] S.-C. Wang, K.-Q. Yan, W.-P. Liao, and S.-S. Wang, "Towards a load balancing in a three-level cloud computing network," in *2010 3rd International Conference on Computer Science and Information Technology*, pp. 108–113, IEEE, Chengdu, July 2010.

[17] A. Singh, D. Juneja, and M. Malhotra, "Autonomous agent based load balancing algorithm in cloud computing," *Procedia Computer Science*, vol. 45, pp. 832–841, 2015.

[18] H. Jinhua, G. Jianhua, G. Sun, and T. Zhao, "A scheduling strategy on Load balancing of virtual machine resources in cloud computing environment," in *2010 3rd International Symposium on Parallel Architectures, Algorithms and Programming*, pp. 89–96, IEEE, Liaoning, China, December 2010.

[19] L. Cherkasova, D. Gupta, and A. Vahdat, "When virtual is harder than real: resource allocation challenges in virtual machine based it environments," Tech. Rep. HPL-25, 2007.

[20] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Virtual Infrastructure management in private and hybrid clouds," *IEEE Internet Computing*, vol. 13, no. 5, pp. 14–22, 2009.

[21] F. Alqahtani, M. Amoon, and A. A. Nasr, "Reliable scheduling and load balancing for requests in cloud-fog computing," *Peer-to-Peer Networking and Applications*, vol. 14, pp. 1905–1916, 2021.

[22] A. A. Nasr, N. A. El-Bahnasawy, G. Attiya, and A. El-Sayed, "Cloudlet scheduling based load balancing on virtual machines in cloud computing environment," *Journal of Internet Technology*, vol. 20, no. 5, pp. 1371–1378, 2019.

[23] M. Iyyappan, A. Kumar, S. Ahmad, S. Jha, B. Alouffi, and A. Alharbi, "A component selection framework of cohesion and coupling metrics," *Computer Systems Science and Engineering*, vol. 44, no. 1, pp. 351–365, 2023.

[24] Iyyappan M, S. Ahmad, S. Jha, A. Alam, M. Yaseen, and H. A. M. Abdeljaber, "A novel ai-based stock market prediction using machine learning algorithm," *Scientific Programming*, vol. 2022, Article ID 4808088, 11 pages, 2022.

[25] A. S. Zamani, M. M. Akhtar, and S. Ahmad, "Emerging cloud computing paradigm," *International Journal of Computer Science Issues*, vol. 8, no. 4, Article ID 304, 2011.