International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.1 (2025) pp. 464-474
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Techniques for load balancing throughout the cloud: a comprehensive literature analysis

Nimmy Francis[1*], N. V. Balaji[2]

[1]Research Scholar, Department of Computer Science, Karpagam Academy of Higher Education, Coimbatore, India
* **Corresponding Author Email:** gardensenimmy@gmail.com- **ORCID:** 0000-0002-0254-5528

[2]Professor, Department of Computer Science, Karpagam Academy of Higher Education, Coimbatore, India
Email: fashdean@kahedu.edu.in- **ORCID:** 0009-0000-0373-4913

Article Info:

DOI: 10.22399/ijcesen.796
Received : 20 November 2024
Accepted : 12 January 2025

Keywords :

Cloud Computing,
Resource Utilization,
Over Load,
Load Balancing,
Fault Tolerance

Abstract:

Recently, "Cloud-Computing (CC)" has become increasingly common because it's a new paradigm for handling massive challenges in a versatile and efficient way. CC is a form of decentralized computation that uses an online network to facilitate the sharing of various computational and computing resources among a large number of consumers, most commonly referred to as "Cloud-Users (CUs)". The burdens on the "Cloud-Server (CS)" could be either light or too heavy, depending on how quickly the volume of CUs and their demands are growing. Higher response times and high resource usage are two of the many issues resulting from these conditions. To address these issues and enhance CS efficiency, the "Load-Balancing (LB)" approaches are very effective. The goal of an LB approach is to identify over-loading and under-loading CSs and distribute the workload accordingly. Publications have employed numerous LB techniques to enhance the broad effectiveness of CS solutions, boost confidence among end CUs, and ensure effective governance and suitable CS. A successful LB technique distributes tasks among the many CSs within the network, thereby increasing performance and maximizing resource utilization. Experts have shown an abundance of engagement on this issue and offered several remedies over the past decade. The primary goal of this extensive review article is to examine different LB variables and provide a critical analysis of current LB techniques. Additionally, this review article outlines the requirements for a new LB technique and explores the challenges associated with LB in the context of CC. Conventional LB techniques are insufficient because they ignore operational efficiency and "Fault-Tolerance (FT)" measures. The present article, to bridge the gaps in existing research, could assist academics in gaining more knowledge about LB techniques within CC.

1. Introduction

Virtualization is a game-changer for CC innovation. As mentioned earlier, there is a need for both software and hardware solutions that can divide physical infrastructure into several virtualization scenarios, each of which may run independently yet share the underlying tangible resources and infrastructure. The integration of CC and Virtualization platforms allows for extensive study in all important disciplines and business applications, as mentioned in several references [1]. The environment of support provided by many "Cloud Service Providers (CSPs)" guarantees consistency, which in turn points to the development of the company at an economical rate, and the automated process is appropriately matched with real CU demands. The "Quality of Service (QoS)" requirements of services delivered via the cloud determine how challenging it must be to provide sufficient resources to these services [2].

Traditional means of allocating resources become ineffective in such an environment due to variation, uncertainty, and resource dispersion. The sudden uptick in interest in these platforms among CUs has inspired the software development industry to create and launch cloud-based, scalable software applications. The system structure that CSPs make accessible is highly dependent on the applications that have been installed [3].

Numerous software developing entities have moved their applications to independent CSP environments

since publicly available on-site sources of resources are limited.

Those in charge of "Data-Centres (DCs)" or CSPs have a responsibility to provide a higher level of availability, performance, and the critical program requirements for growth [4].

By distributing tasks across all available units for processing, publicly accessible cloud load distribution ensures that all computing resources are used to their full potential. The LB within CC typically is required, but in a more particular sense [5]. There's a pressing requirement for LB within CC due to the following issues: the CU demands have been multifaceted, the flow of network traffic to a CSP is undetermined and non-probability, there's not a reliable resources assigner for CU inquiries, jobs are not distributed throughout resources for computing, which includes dependence, and demand for resources fluctuate according to CU demands. LB uses a network of nodes for dispersing the load [6]. One of the main goals of LB techniques is to choose tasks in a way that reduces processing time as well as resource usage within DCs. Every aspect of computational resource virtualization is part of the enormous new field that has emerged to investigate and create in response to the compelling needs of CC. Since CC provides an establishment that concentrates on CUs, genuine CUs could increase their revenue by concentrating on their numerous operations and using an appropriate allocation of resources [7]. In a typical CC LB process, there are two stages. In the beginning, at the hierarchy of "Physical-Machines (PMs)" (in which job migration occurs in two distinct phases, inter-VM, and intra-VM), the LB handles and allocates the workload within the corresponding "Virtual-Machines (VMs)" (in which each of the VMs uses different LB methods) [8]. Several resources are needed to execute the CU tasks known as Request-Generators, and these in turn produce CU demands. Figure 1 shows how the DC controller manages tasks.

Problem Statement: An increasingly significant challenge in this context, the "Task-Scheduling (TS)" concern is driving up costs as the volume of CSPs as well as the workload on the CSs continue to rise. Certain VMs can be largely utilized whereas others stay insufficiently utilized when running TS on them. Hence, an effective LB is necessary to arrange the TS operation and distribute the workload on CSs evenly [9]. LB maintains a reputation for being an effective way of distributing the load across all of the VMs within an environment. By using this approach, the developers can be confident that every single VM handles about the same amount of work. It eliminates load imbalances, that could lead to network delays, and increases outcomes like speed, speed of response, dependability, and usage of resources. For many causes, the LB within the CC context might malfunction or terminate. The network becomes unavailable and CC's reputation is diminished as a result. There are typically 4 primary areas where CC failures might manifest: amongst CSPs, within CSPs, over CSPs alongside CUs, and within CUs themselves. Additional energy usage and financial losses might result from CSP breakdowns. The time taken to respond for necessary services may be increased if CUs collapse [10]. A crucial and essential aspect of CC involves FT. This allows the infrastructure to detect the nature and precise spot of the problems and strive to tolerate that, allowing cloud-based services to be offered even when faults are present.

Paper Contribution: Multiple scholars in the last ten years have investigated LB approaches in CC settings, providing a firm groundwork for comprehending the multiple facets of this problem. Based on this research and observations, it seems that the scientific literature lacks a comprehensive and well-structured analysis of the existing LB approaches. Consequently, this article aims to fill this void by providing a structured and comprehensive examination of current LB approaches through the adoption of a methodical strategy. Additionally, this study details the successful initiatives in this area, compares them in depth, identifies difficult challenges, as well as finally, suggests ways ahead for research in this particular domain.

To sum up, the primary objectives of this research are:

- Making it clear ways to apply an organized approach to this area of study.
- Researching and classifying LB tactics into 3 broad categories: Static-LB, Dynamic-LB, and Hybrid-LB, while outlining the benefits and drawbacks of each.
- Bringing attention to the challenging issues and unanswered questions in this area to enhance prior experiments.

Paper Organization: Section 2 provides a survey of the current LB review published in CC, while Section 3 reviews the methodologies of LB along with its terminologies and unveils a few chosen LB techniques. Section 4 compares and contrasts the reviewed methods, identifies unresolved problems, and suggests possible future developments and the last Section 5 concludes this review article.

2. Related works:

The researchers classify and taxonomies TS articles according to several LB techniques for CC in [11].

Along with some limitations and unanswered questions, the benefits and drawbacks of the LB methods have also been detailed. However, they neglected to discuss the LB computational strategies, records, assessment methodologies, or evaluation with other tools used for simulation.

Researchers in [12] provide a thorough analysis of LB along with TS techniques within CC, along with an ordering and categorization of multifaceted systems. By outlining the problems and unanswered questions, they were able to assess the findings of the relevant studies that were selected. Nevertheless, this review did not include the following: the process for selecting articles, the time frame of the chosen research, the systems' characteristics, assessment methods, or comparisons of the tools used for simulation. Following outlining the standards for inclusion and exclusion, the researchers of [13] examined 56 papers about fog-based TS. An overview of the research was provided by noting the examined LB approaches along with their merits and drawbacks. Additionally, new initiatives and research limitations in current solutions are also recognized. On the other hand, assessment techniques haven't been mentioned. The researchers in [14] examined the CC's meta-heuristic methods for performing LB. They classified and examined various LB methods and taxonomies, as well as the benefits and drawbacks of various procedures. In addition to comparing commonly employed simulation programs, they additionally emphasized open concerns and potential developments. However, the research does not take algorithmic techniques or database assessment into account.

In their most recent analysis, the authors in [15] examined Fog-Computing along with the “Internet-of-Everything (IoE)”. After introducing the various optimizing measures, they went on to categorize the various LB approaches that are now available. To find unanswered questions and potential avenues for further research, the papers were reviewed. There was an omission of information on the length of the research and the procedure for selecting the articles. Another omission from the survey was information on the LB computations, databases, assessment techniques, and simulation instruments used.

3. Methodologies

3.1 Process of LB in CC

The main goal of LB remains to make sure that none CS was under-load or over-load by properly managing the load throughout each of the CSs. One way to define LB is the practice of distributing a load across many devices or networks of systems to make the most efficient usage of available resources and get the best possible total speed of response. As an added advantage, it keeps resources from being duplicated too much and shortens the device's overall waiting time. With this procedure, requests are dispersed within CSs to share and handle information with no waiting. By shifting the load from one device to another, LB enhances the system's efficiency. Figure 2 depicts the LB process within CC. LB offers a methodical approach to distributing tasks evenly across the available resources. During the case of a service outage, the objective is to continue providing dependable service through provisioning and de-provisioning the system instance, while also making sufficient usage of the resources. Furthermore, LB's goal is to improve the efficiency of resources and decrease the time for responding, leading to lower-cost, higher-efficiency devices.

3.2 Types of LB

There are three types of LB in the CC context. They are detailed in the following section:

Static-LB (SLB) Algorithms:

Before the actual running time of the tasks starts, the SLB mechanisms disperse them among VMs. Such techniques lead to inefficient use of resources since they load certain VMs while they are operating. Resources consumption is increased and the number of "Service-Level-Agreement (SLA)" breaches due to overloaded VMs is increased by SLB techniques. Here are a few primary SLB approaches:

Round-Robin (RR): The smallest job is chosen and done firstly according to this SLB technique. By reducing task waiting periods and hence preventing deprivation, the shortest possible task strategy improves the cloud's efficiency and gives computations an edge against rival techniques. After randomly selecting the hub, each node shall receive its workspace distributed to it within an RR fashion.

A benefit for DCs is that every single VM shares identical processing capability. Task length, resource significance, and capacity are not considered in this RR method. Regardless, it's possible that certain resources are being overutilized, and it's also common for other resources to sit idle.

Min-Min: All task-related information is accessible from the prior phase of this SLB method. A list of all unfinished tasks is the starting point for this method. The amount of time required to do each activity was calculated using this method.

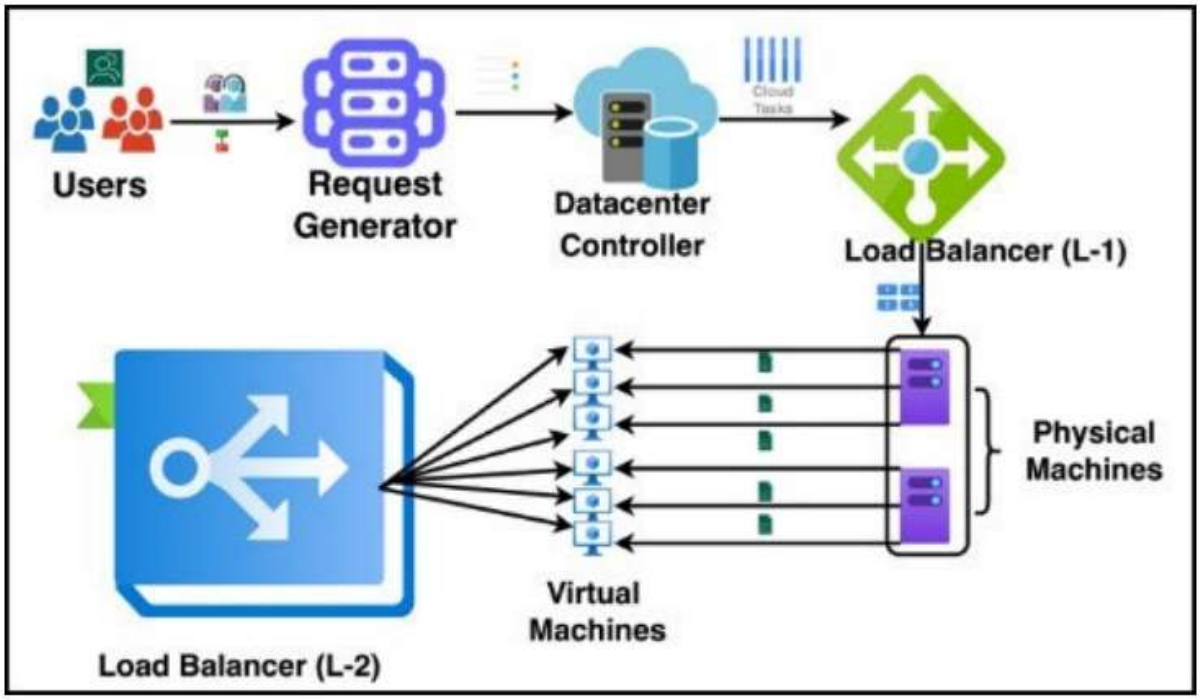

Figure 1. *Overall LB system*

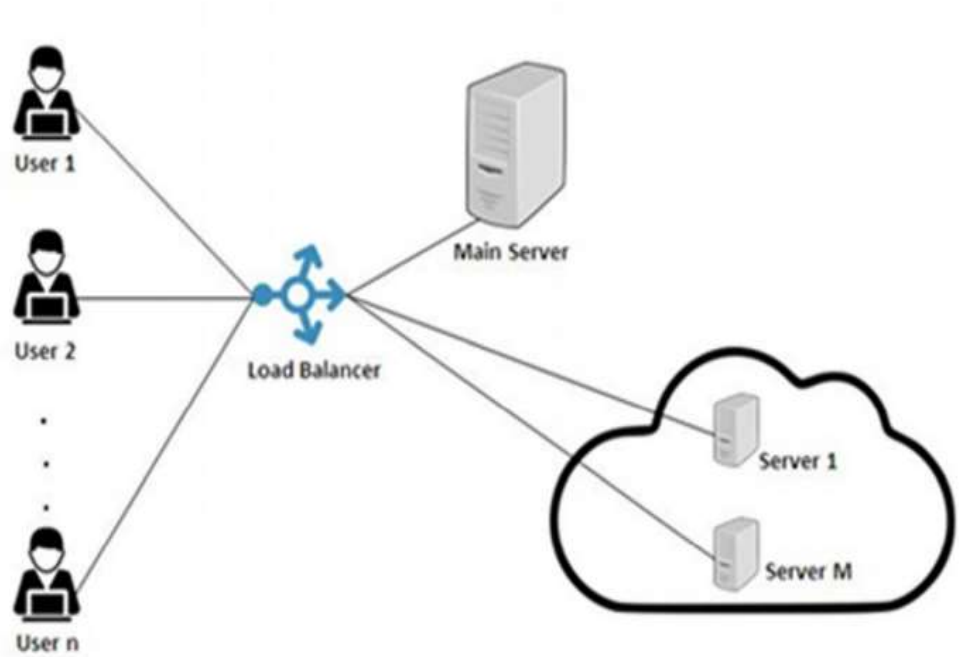

Figure 2. *The LB process within the CC*

For every job, it selects the bare minimum of time needed to finish. Out of all the tasks, it selects the resource that takes the shortest time to finish. Finally, this method creates a map for every job that has been chosen. Once all non-allocated tasks have been designated, this function terminates. The most time-efficient task gets completed first using this method, which can be an enormous advantage. One potential downside of this strategy is the fact it could result in the elimination of certain tasks.

Max-Min: When contrasted with the min-min technique, this SLB methodology employs the inverse strategy.

Here the highest priority tasks are carried out initially. It works very comparable to the Min-Min method, with the exception that this approach chooses the highest priority after finding the jobs with the shortest time for completion.

Dynamic-LB (DLB) Algorithms:

Throughout the execution of the task, the DLB method updates the workload of VMs according to the system's status and allocates accepted tasks to

them. In an attempt to circumvent the shortcomings of the SLB approaches, DLB disregards the prior data about the system's status. Complexity abounds in the DLB methods. In contrast to SLB methods, nevertheless, these methods work better and have higher FT.

Statistics-DLB Approaches: For optimal workloads and to assess the system's efficiency, these techniques are put into effect. Several QoS-orientated methods have been suggested to enhance the usage of resource efficiency.

Swarm-Behaviour Approaches: Two separate branches of these approaches draw their inspiration from smart biological processes: nature-oriented and nature-inspired. Several "Swarm-Intelligent (SI)" methods were recently suggested for DLB on the cloud.

Evolution-Based Approaches: These strategies have been demonstrated with the most sophisticated and expansive search spaces within the particular DLB techniques.

Hybrid-LB (HLB) Algorithms:

The characteristics of HLB strategies are a mix of those of DLB along SLB techniques. Its purpose is to ensure that CC features are always available to CUs.

3.3 Different Types of Current LB Techniques

Current Techniques under SLB

The Standard-LB methods come under the SLB category. In this section, some of the most used SLB was reviewed as follows:

Randomized-Technique: It is not necessary to be aware of the VM's current or historical load when attempting to implement this technique's recommended choice of VMs. This works well when each VM takes on the same amount of work from the system. Its static functioning is the reason for this.

Threshold-Technique: After the creation of one VM, the workload can be promptly distributed using this SLB-oriented approach. Despite transmitting any controlling codes, the choice of VM is done immediately. VMs save one copy from the load data. Min-load, Med-load, and High-load are the three main types of load characterization. Whenever a query reaches the CS, the BS correlates the total of HTTP requests with the specified threshold level. The workload condition goes into overdrive when the current CU query frequency is higher than a predetermined threshold. At the same time, things are looking despairing for the workload.

Opportunistic-LB Technique: While it's an SLB method, the modern VM workload is not specified anymore. It intends to keep each VM busy. It’s an effort to keep all hubs active despite its lack of knowledge of the present burden of each system. There is an open-source program available for this method, and it can display every transaction to a support department. Every application is managed by the CS according to a predetermined timeline. The approach is tailored to comply with the requirements set forth by the CU for the particular procedures. Following that, intending to distribute the workload evenly across the several CSs, cloud-based DCs implement the opportunistic LB method.

Throttled-LB: This method relies on the LB keeping track of the availability and use of VM indexes within a database. To complete the assigned task, the CU asks the DC to locate an appropriate VM. Because of the VM, the DC needs an LB to manage it. Before allocating VMs to the LB, the DC must execute a query. With the unlikely scenario that not enough VMs were found, the LB would head back to the DC to execute the CU's request. Therefore, for the task to begin, the CU will notify the LB to figure out the best VM to carry out the required tasks.

Stochastic Hill-Climbing: A mediocre strategy for problem-solving, this methodology is a variation of the hill-climbing methodology. Because the dispersed LB mechanism shown herein eliminates the obstacle, improving the distribution of system workloads has additionally been considered. The route of giant-value is characterized by a similar loop, which travels continually upwards or backward. This halts when a neighbor with a higher curiosity level is detected.

Equally-Spread Current-Execution: Every VM was given priority for a certain task because of the equitable distribution of tasks. The Spreading-Spectrum technique is used here, and it involves distributing the load among several VMs according to the load intensity. The LB assigns the task to the appropriate VM along with a little tweak to its hardware, which increases the VM's output. One aspect of this method is that it keeps track of VMs also the volume of requests that are presently assigned to them. VMs commence with no allocations. It fails to allocate TS whenever there are more requests than expected.

Join-Idle Queue (JIQ): Persistent online services and large-scale shared infrastructures are both made possible by it. The pooled dispatchers represent how this method functions. By excluding the LB service from the primary route, this technique forces the optimum processor to notify the dispatcher about its inactivity despite the absence of task expertise regarding the application. All of JIQ's pooled planners have an I-queue which stores a group of CSs that aren't actively working. Over random intervals after joining the network, newly

arrived objects encounter schedulers and request to be added to the idle method's I-queue.

Listing-Tasks: Numerous processes share the available processing power as they execute in parallel on several many-core devices used for computing TS. It is difficult to optimize system efficiency and usage of energy by allocating cores to various activities. Considering time and energy restrictions further complicates matters. A method for determining before and post-power for parallel processes that rely on lists was suggested, and it works on both constant and periodic speed durations. The method reduced calculation costs and make-span. On the other hand, the transmission costs remained larger.

Task-Priority: Recent developments in CC-based IoT have made it possible for the creation of lower-latency applications necessitating immediate feedback. The VM which could do this job in the least amount of time was used to complete it. The use of multiple queues allowed it to resolve the starving of lower-priority activities with resources that were idle, which improved make span, reaction time, usage of resources, and throughput. Although this method cuts down on waiting and task completion times, it was unable to accomplish LB in a parallel fashion across several VMs.

Earliest-Deadline First (EDF): When it comes to time-sensitive bag-of-tasks, the majority of solutions overlook the ever-changing nature of clouds. Considering a distributed setting, an EDF-based scheduling method enhanced the volume of time constraints met while reducing infrastructural expenditure. The goal of developing this method was to efficiently complete the tasks at hand while keeping costs to a minimum. Whenever the processing length of tasks got higher, however, responding was prolonged.

Current Techniques under DLB

It's difficult to solve the issue of dynamic TS across a diverse context. In this section, some of the most used DLB techniques were reviewed as follows:

Particle-Swarm Optimization (PSO): These optimization techniques are very clever and mimic the swarm-oriented behavior of animals. They are called bionic-heuristic methods. To better manage jobs and distribute the load, a framework for energy usage was also included, along with an enhanced PSO technique. Task execution times were additionally lowered with a low-complexity binary variant using the PSO technique. By using its strengths in versatility, easy recognition, great resilience, and exceptional performance in dynamic contexts, PSO effectively resolves several problems associated with paired optimization. When it comes to fixing problems with differential restrictions, the PSO method is inefficient.

Genetic-Algorithm (GA): Using principles from genetics and selective breeding, it constitutes a stochastic exploration method. A basic GA with three activities: "Availability", "Genetics", and "Replacing". At its foundation is the process of creating new generations through mutations and crossovers, predicated on the idea that different types of chromosomes need different types of coding: "Binary-Coding", "Tree-Coding", or "Numerical-Coding". When trying to recreate the actual process of genetic processes and the idea regarding biological evolution put forward by Charles Darwin, GA becomes the preferred natural computation methodology. It makes an effort to decrease the completion time of the tasks listed in the waiting list simultaneously balancing the load on the cloud's resources. It shortened the time while increasing the size of process applications along with the error frequency. However, it requires more time on computational capabilities to handle huge solutions.

Ant-Colony Optimization (ACO): The driving force of this model is a cooperative group of ants that, over many years, seeks to discover the optimal manner to connect various locations, share information through the sharing of pheromone signals (like naturally occurring food-seeking ants), and examines multiple options into a suitable variable or topological data storage. In CC, the ACO in LB manages the load and cuts down on makespan. Every task is thought to be highly computational and distinct from one another. In addition, a plan for scalability was developed to enhance TS and energy savings. Improving ACO's convergence speed despite sacrificing solution variation is possible by extending its linear Weighted-Sum. Getting into regional optimum problems the outputs can be accomplished when dealing with the problem of large-scale optimization that leads to poor algorithmic efficiency.

Honey-Bee Foraging (HBF): Following the lead of honeybees, this method maximizes output by adjusting the amount of nectar (speed). Throughout living in the colony, it is believed that they've got a variety of duties to play. Bees that engage in Active-Foraging search for food sources, then return to the hive after gathering food. The scout-bees actively investigate their environment in search of fresh sources of food, which they are accumulating on a VM table. Some of those foraging-bees stop moving around at certain points in time. This method is a kind of foraging that could be included in the LB strategy for TS operations within the CC. Results for autonomous

TS were closer to ideal as a result of its quicker convergence. With almost elevated CU request volumes, yet, the delay becomes a significant issue.

Whale-Optimization (WO): TS keeps getting increasingly complicated when the quantity of assignments and resources increases. For optimal energy usage and time, the WO approach is used as an LB to TS. The main emphasis of this WO approach is to mimic the way whales behave. While opposition-based training has several advantages, it may be operationally costly to evaluate all of the possible alternatives. Additionally, it is noticeable how Chaos-Theory increases the WO's overhead. There will be a noticeable change to the algorithm's complexity in time with even a little rise in its parameters. The problems caused by the method's excessive processing complexity may be reduced by parallelization.

Simulated-Annealing (SA): SA is inspired by solid-state annealing, which entails heating and gradually cooling a material (such as glass or metal) to remove and compress its internal tensions. The process additionally becomes sensitive to CU request frequency and bin quantity; therefore, it is common for this approach to be frozen by local restrictions and generate unwanted VM allocations. Using a goal-programming strategy, the SA in LB for CC reduces operation costs and delays. Latency duration, timeline satisfaction, and accessibility level constraints were all enhanced using a "Multi-Objective SA (MOSA)" technique. Through working together, they succeeded in accomplishing several objectives, along it adding the aim of accessibility level to better divide up CC work. Still, this method came at a huge expense in terms of communication.

Cuckoo-Search Optimization (CSO): A meta-heuristic approach, the CSO method simulates the actions of cuckoo birds in their native environment. This approach finds the optimal solution while balancing local and global analysis through the support of parameter flipping. The results obtained surpass those of the PSO.

(viii) Osmosis-LB (OLB): Ultimately, this OLB approach aims to modify the load by reassigning work to a succession of VMs. Decentralized Chord-Overlays run by agents similar to ants are the backbone of this LB mechanism. Every DC has its unique implementation-specific features, communicates with a list, and may do many jobs concurrently.

Tabu-Search Optimization (TSO): As a global optimum approach with a higher-grade optimizing capacity, this TSO methodology aspires to mimic human cognition. When utilized for problems like allocation of resources and optimized performance, it aims to point other methods away from falling into the local optimum dilemma.

Reinforcement-Learning (RL): The massive amounts of information produced by IoT gadgets do not just overwhelm the CC system, additionally, they cause latency to rise as a result of the large number of hops. Because it impacts processing speed and causes system crashes, using a VM that is either too heavy or too light is consistently not an option. An RL approach is used to include scheduled upkeep within a diverse setting, enhancing FT with the least cost. Two dynamic programming-oriented issues are presented. The issues have been transformed into RL-approximated computational challenges. It enhanced scheduling performance while decreasing the computationally demanding nature of the challenges. On the other hand, the tasks' latency remained disregarded.

Current Techniques under HLB

The researchers suggested the hybrid approach in [16] by combining PSO with "Fuzzy-Logic (FL)" along with SA, resulting in FL-PSO and SA-PSO, accordingly. TS was made even more effective by combining the suggested methods alongside the dynamic response queues methodology. Especially for higher-dimensional situations, the innovative methods were shown to be beneficial. This research aimed at optimizing make-span, expenses, LB, time spent waiting, usage of resources, and length of queue simultaneously. However, the research did not account for the delay.

For TS separate activities without causing them to converge too soon, the researchers of [17] suggested a hybrid technique that optimized the “Dragonfly-Algorithm (DA)” using a combination of the “Mexican-Hat Wavelet-Transform”, and a “Biography-based Optimization-Migration” procedure. Time to respond, operation, and SLA breaches had all been improved by dynamically scheduling the activities. Mutation process performance may have been much better with the correct weights.

Regarding a combination of approaches for TS, the researchers suggested an updated “Henry-Gas Solubility-Optimization (HGSO)” that employs WOA and “Comprehensive-Opposition-Based Learning (COBL)” [18]. Local searching was enhanced by the WOA, while the worst-case scenario was mitigated by the COBL. By comparing it to WOA, HGSO, MFO, FA, PSO, and SSA on both artificial and actual databases, the suggested method outperformed its competitors in terms of make-span and efficiency. However, every method that was compared was the most basic form.

In their proposal for a hybrid scheme, the researchers of [19] used opposition-based learning and "Differential-Evolution (DE)" to create a revised "Fire-Works Algorithm (FWA)". DE eschewed local optimal solutions, but opposition-based learning enabled a diverse collection of alternatives. By reducing make-span and expense while enhancing the usage of resources, the approach achieved effectiveness. Having said that, the researcher claims that the suggested method is not resilient to node outages and task transfer.

The researchers suggested a hybrid model called "Hybrid-Firebug and Tunicate-Optimization (HFTO)" that may improve LB and reduce the duration it took to complete tasks [20]. Various VMs were put together and the loads have been organized according to their characteristics. It's improved speed, reaction time, make-span, and FT as well. Workloads were distributed to VMs according to their peak needs. VMs that had minimal CPU usage received lighter tasks, whereas those that had substantial CPU usage were given computation-intensive ones. Despite having relatively few resources, the method managed to enhance both make-span and complexity of computation. The workload that had been simulated, however, remained undisclosed.

4. Discussion of the study

4.1 Limitations of LB within CC

Among the several difficulties that CC must overcome, LB stands out as an extremely pressing issue that requires immediate action. Finding an improved way of using cloud resources requires balanced consideration for concerns which include migrating VMs, privacy of VMs, QoS by CS, and the efficient use of resources. The following are some LB concerns:

Geographically Distributed VMs: Most DCs of the CSP have been distributed so that CUs may access them from anywhere. To efficiently handle CU requests, the dynamically dispersed VMs within those DCs act like a consolidated network. While there are several LB methods out there, they all have their limitations and fail to account for important factors like network and communication delays, VM spectrum, CU space, and resources that are accessible. Some techniques aren't well-suited to very distant locations, making it difficult to run VMs there.

Isolated Failure Point: Particular LB techniques have been suggested by researchers for cases where a single consolidated VM makes LB choices rather than a distributed set of VMs. A computer system as a whole is vulnerable to failures in critical components.

Migrating VMs: It is possible to construct several VMs upon a single PM by using virtualization. Such VMs were independent in design and had diverse configurations. It's reasonable to use an LB approach to move every VM to a distant place if a PM is overwhelmed.

Diverse VMs: As part of their preliminary investigation, the writers have suggested using uniform VMs throughout LB. The CUs of CC require an adaptive switch, that can only be implemented on diverse VMs to achieve network efficiency and decrease reaction time.

Data Management: CC solved the problem of traditional storage systems, which were expensive and resource-intensive. There are no control difficulties when CUs maintain information heterogeneously within the cloud. Duplicating saved information is essential for optimal accessibility as well as information continuity since storage continues to grow at an exponential rate.

Scalability of LB: The capacity to quickly scale up or down is only one of the many benefits of using cloud-based services, which also provide on-demand scaling. When designing a robust LB, keep in mind the ever-evolving demands of computing, storage, device topological structure, etc.

Level of Method's Complexity: Methods for CC need to be easy to implement and work quickly. Improving the cloud's effectiveness and quality is the goal of a resilient algorithm.

Autonomous provisioning of services: The ability to autonomously allocate or assign resources constitutes a crucial feature of CC. The question thus becomes how to make use of or withdraw from cloud-based services while preserving an identical level of productivity as traditional systems while making optimal use of available resources.

Energy Efficiency: One advantage regarding energy administration is the usage of the cloud, which promotes economy on the scale. Reduced energy consumption is the single most critical factor that will enable an international market to function in which publicly traded enterprises contribute to a common fund instead of competing privately.

4.2 Future Paths of Research

These findings show that there are still many open questions in this field that require further investigation. The research found ways to improve the LB methods for future studies, which would optimize the CC operations. Here are some of them:

- The majority of studies take place in a simulated setting; numerous techniques are being

developed to simulate the CC setting, however putting them into action in real-time can be quite difficult; hence, relatively limited techniques have found practical use in this setting. When developing techniques that might encounter problems in real time, it is possible to take advantage of open-source infrastructures like Open-Stack and Cloud-Foundry.

- Rapidly vertically growing CSs within the cloud DC, tasks with high priority due to be executed, shifts in processes, machine setups, and other circumstances could all complicate the technique. To address these concerns and enhance scaling, efficiency, and speed, an effective structure is necessary.
- Due to its periodic nature and lack of data storage, scheduling-aware LB techniques like Round-Robin cause an unequal allocation of workload across VMs.
- To make processes that are both lightweight and capable of overcoming hybridization issues, it is possible to combine SLB methods with DLB methods which are influenced by nature or alternative methods; however, this merges the system's complexity.
- With the success of CC usage in healthcare, the community stands to gain from the development of a health-specific, optimal LB mechanism.
- Both TS and LB issues are thought of as being NP-hard. While methods that draw inspiration from nature are making significant progress, there remain numerous unresolved issues. For example, how to create the best fitness-function to accurately assess potential resources and guarantee scalability for activities with changing resource needs.
- Following a regional LB process, QoS-based LB ensures an appropriate level of service regardless of whether resources are available or not. However, when trying to overcome the aforementioned obstacles, results in substantial management of overheads that need thorough scenario assessment.
- While suggesting an efficient approach for use in a cloud setting that processes data in real-time, the FT variable is often disregarded by many existing LB approaches.

5. Conclusions

A crucial component of the CC field, LB serves to enhance the allocation of workloads and the administration of resources, to minimize the system's overall reaction time. Numerous methods and techniques are being developed to handle LB-related issues, including allocating resources, migrating tasks, and optimizing the utilization of resources. Various approaches within the LB adhering CC field were investigated in this present research. Researchers have looked at the problems with LB and studied the solutions thoroughly in the past few decades. Problems with the migration of VMs and FT issues, among others, persist within the CC context despite the presentation of several solutions. With the help of this literature review, researchers have a lot of space to create cutting-edge LB approaches that work well in CC settings. A large portion of the survey dedicates itself to discussing the SLB, DLB, and HLB methods. Academics may find this investigation useful for discovering LB-related research difficulties, such as how to reduce reaction time and prevent VM failures, as it includes a review of present and previous LB methods. In addition, it made specific recommendations for future studies that should lead to the development of an ideal LB method that can address all problems with current LB techniques and work in a real-world CC setting. Cloud Computing has been studied and reported in the literature [21-28].

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1]S. Mangalampalli, (2023). Cloud computing and virtualization, *Convergence of Cloud with AI for Big Data Analytics: Foundations and Innovation*, 2023, pp. 13–40.

[2]A. Sunyaev, (2020). Cloud computing, *Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies.* 2020, pp. 195–236.

[3]X. Fu, Y. Sun, H. Wang, and H. Li, (2023). Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm, *Cluster Comput.*, 26(5);2479–2488.

[4]M. H. Shirvani, (2023). An energy-efficient topology-aware virtual machine placement in cloud datacenters: A multi-objective discrete Jaya optimization, *Sustain. Comput., Informat. Syst.*, 38,100856.

[5]S. Mangalampalli, G. R. Karri, M. Kumar, O. I. Khalaf, C. A. T. Romero, and G. A. Sahib, (2024).DRLBTSA: Deep reinforcement learning based task scheduling algorithm in cloud computing, *Multimedia Tools Appl*., 83(3);8359–8387.

[6]H. Mikram, S. El Kafhali, and Y. Saadi, (2024) HEPGA: A new effective hybrid algorithm for scientific workflow scheduling in cloud computing environment, *Simul. Model. Pract. Theory*, 130;102864.

[7]S. Iftikhar, M. M. M. Ahmad, S. Tuli, D. Chowdhury, M. Xu, S. S. Gill, and S. Uhlig, (2023) HunterPlus: AI-based energy-efficient task scheduling for cloud–fog computing environments, *Internet Things*, 21;100667.

[8]P. Pirozmand, (2023). An improved particle swarm optimization algorithm for task scheduling in cloud computing, *J. Ambient Intell. Humanized Comput.*, 14(4);4313–4327,

[9]J. Elcock and N. Edward, (2023). An efficient ACO-based algorithm for task scheduling in heterogeneous multiprocessing environments, *Array*, 17;100280.

[10]Y. Cheng, Z. Cao, X. Zhang, Q. Cao, and D. Zhang, (2023). Multi-objective dynamic task scheduling optimization algorithm based on deep reinforcement learning, *J. Supercomput.*, pp. 1–29.

[11]A. Amini Motlagh, A. Movaghar, and A. M. Rahmani, (2020). Task scheduling mechanisms in cloud computing: A systematic review, *Int. J. Commun. Syst.*, 33(6);e4302.

[12]M. Hosseinzadeh, M. Y. Ghafour, H. K. Hama, B. Vo, and A. Khoshnevis, (2021). Multi-objective task and workflow scheduling approaches in cloud computing: A comprehensive review, *J. Grid Comput.*, 18(3),327–356

[13]N. Kaur, A. Kumar, and R. Kumar, (2021). A systematic review on task scheduling in fog computing: Taxonomy, tools, challenges, and future directions. *Concurrency Comput., Pract. Exper.*, 33(21);e6432.

[14]E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, (2021). Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends, *Swarm Evol. Comput.*, 62;100841.

[15]B. Jamil, H. Ijaz, M. Shojafar, K. Munir, and R. Buyya, (2022). Resource allocation and task scheduling in fog computing and Internet of everything environments: A taxonomy, review, and future directions, *ACM Comput. Surveys*, 54(11);1–38.

[16]H. Ben Alla, S. Ben Alla, A. Touhafi, and A. Ezzati, (2018). A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment, *Cluster Comput.*, 21(4);1797–1820

[17]M. R. Shirani and F. Safi-Esfahani, (2021). Dynamic scheduling of tasks in cloud computing applying dragonfly algorithm, biogeography-based optimization algorithm and Mexican hat wavelet, *J. Supercomput.*, 77(2);1214–1272

[18]M. A. Elaziz and I. Attiya, (2021). An improved Henry gas solubility optimization algorithm for task scheduling in cloud computing, *Artif.Intell. Rev.*, 54(5);3599–3637.

[19]A. M. Yadav, K. N. Tripathi, and S. C. Sharma, (2022). An enhanced multi-objective fireworks algorithm for task scheduling in fog computing environment, *Cluster Comput*., 25(2);983–998.

[20]M. Nanjappan, G. Natesan, and P. Krishnadoss, (2023). HFTO: Hybrid firebug tunicate optimizer for fault tolerance and dynamic task scheduling in cloud computing, *Wireless Pers. Commun*., 129(1);323–344.

[21]Iqbal, A., Shaima Qureshi, & Mohammad Ahsan Chishti. (2025). Bringing Context into IoT: Vision and Research Challenges. *International Journal of Computational and Experimental Science and Engineering*, 11(1). https://doi.org/10.22399/ijcesen.760

[22]M. Revathi, K. Manju, B. Chitradevi, B. Senthilkumaran, T. Suresh, & A. Sathiya. (2025). Metaheuristic-Driven Optimization for Efficient Resource Allocation in Cloud Environments. *International Journal of Computational and Experimental Science and Engineering*, 11(1). https://doi.org/10.22399/ijcesen.831

[23]YAKUT , Önder. (2023). Diabetes Prediction Using Colab Notebook Based Machine Learning Methods. *International Journal of Computational and Experimental Science and Engineering*, 9(1), 36–41. Retrieved from https://ijcesen.com/index.php/ijcesen/article/view/187

[24]S.P. Lalitha, & A. Murugan. (2024). Performance Analysis of Priority Generation System for Multimedia Video using ANFIS Classifier. *International Journal of Computational and Experimental Science and Engineering*, 10(4). https://doi.org/10.22399/ijcesen.707

[25]BENTAJER, A., HEDABOU , M., ENNAAMA, F., & ELFEZAZİ , S. (2020). Development of Design for Enhancing Trust in Cloud's SPI Stack. *International Journal of Computational and Experimental Science and Engineering*, 6(1), 13–18. Retrieved from https://ijcesen.com/index.php/ijcesen/article/view/109

[26]Pattanaik, B. C., Sahoo, B. kumar, Pati, B., & Pradhan, A. (2024). Enhancing Fault Tolerance in Cloud Computing using Modified Deep Q-Network (M-DQN) for Optimal Load Balancing. *International Journal of Computational and*

Experimental Science and Engineering, 10(4). https://doi.org/10.22399/ijcesen.601

[27]Sankari, A. S., & S. Vimalanand. (2024). Biased Random Sampling with Firefly Optimization (BRS-FO) based on Load Balancing for Virtual Machine Migration in Cloud Computing. *International Journal of Computational and Experimental Science and Engineering*, 10(4). https://doi.org/10.22399/ijcesen.753

[28]V. Ananthakrishna, & Chandra Shekhar Yadav. (2025). QP-ChainSZKP: A Quantum-Proof Blockchain Framework for Scalable and Secure Cloud Applications. *International Journal of Computational and Experimental Science and Engineering*, 11(1). https://doi.org/10.22399/ijcesen.718