Smart Internet of Things

www.siot.reapress.com

Smart. Internet. Things. Vol. 1, No. 1 (2024) 54–66.

Paper Type: Original Article

Balancing the Load: An Evolution Story of Load Balancing Algorithm

Anjali Priya[1,*], Ipsit Chandra[1], Debdatta Pal[1], Chinmay Tiwari[1]

[1] Department of Computer Engineering, KIIT University, Bhubaneswar-751024, Odisha, India; 21051796@kiit.ac.in, 21051056@kiit.ac.in, 21051048@kiit.ac.in, 21052246@kiit.ac.in.

Citation:

Received: 16 January 2024 *Revised: 17 March 2024* *Accepted: 19 June 2024*	Priya, A., Chandra, I., Pal, D., & Tiwari, Ch. (2024). Balancing the load: an evolution story of load balancing algorithm. *Smart Internet of Things, 1* (1), 54-66.

Abstract

Load Balancing (LB) is important in cloud computing for managing the cloud resources efficiently. It involves distribution of incoming network traffic and workload among various servers so that no single server is overloaded. This results in improved resource utilization, increased throughput, and decreased response time. LB is very important for cloud systems to achieve high availability and fault tolerance. LB has undergone various transformations to meet the demands of modern applications. This research aims in highlighting the history of evolution of LB algorithm, tracing their journey from the static methods to dynamic and adaptive approaches. This study informs about the principles of LB and the challenges faced by the traditional algorithms. There are different kinds of load-balancing algorithms that differ in terms of their complexity, flexibility, and performance. Load-balancing algorithms play a crucial role in ensuring the smooth operation of modern computing systems.

Keywords: Load balancing, Distributed system, Server load, Scalability, Optimization, Load balancer algorithm, Dynamic load balancing.

1 | Introduction

Cloud computing revolutionizes how companies and individuals' access and uses computing resources by delivering services like storage, processing power, and software over the Internet. This technology saves time and reduces costs associated with installation and configuration. Its popularity among researchers has grown, particularly in genomics, machine learning, and big data analytics, enabling innovative discoveries. Scalability is a key benefit, allowing researchers to rapidly scale up resources without expensive infrastructure [1]. Cloud Computing promotes collaboration and sharing of resources thus simplifying many extensive projects. The main service models of cloud computing are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). IaaS providers include Amazon Web Services, Microsoft Azure, and Google

Corresponding Author: 21051796@kiit.ac.in

Cloud Platform. Despite its advantages, challenges like data security and LB exist. Resource scheduling and load-balancing algorithms are crucial for efficient resource utilization in cloud computing systems, contributing to effective task allocation, workload distribution, and overall system performance [2]. Various load-balancing algorithms, such as IP Hash, Least Connections, Round Robin, Genetic Task Scheduling Algorithm, Particle Swarm Optimization (PSO), and Flower Pollination Algorithm (FPA), play a vital role in managing incoming traffic and optimizing resource usage. *Fig 1* depicts the structure of LB.

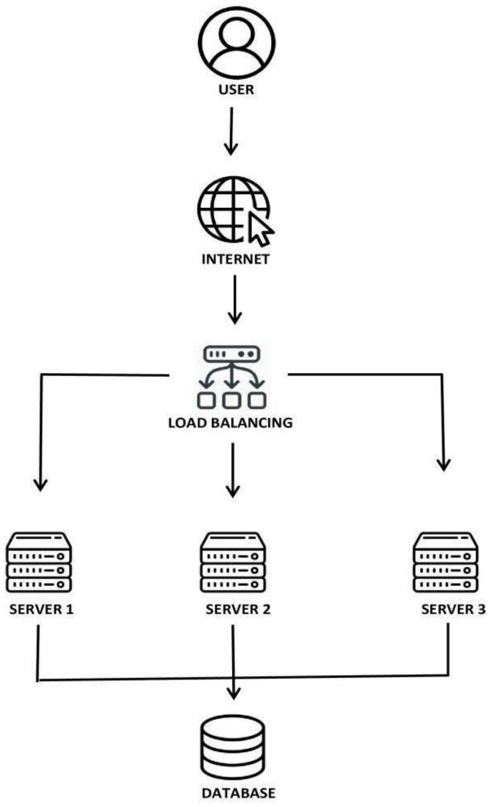

Fig. 1. Load balancing.

1.1 | Variables and Equations

This paper traces the evolution of LB algorithms, based more on their principles and applications, rather than on mathematical formulations. However, there are some concepts and metrics that hold the key to performance and effectiveness [3]. Server load: the document does not say what a certain parameter is, so it basically means the task load on the different servers, in order that the server does not remain high loaded compared to the other servers. Task allocation: it refers to the assignment of incoming tasks to various servers in such a way that it assures optimal use of resources. Algorithm efficiency: the effectiveness of the algorithm is actually generally determined to be the capacity to reduce reaction time in addition to making the most efficient usage of resources. Despite not being articulated with a specific equation in the document, efficiency is articulately central to the issue [4].

Conceptual equations, as much as they are not specific, the underlying principles must be framed in a general sense and stated as follows:

I. LB objective

$$\text{Minimize max } (L_{\text{server}}). \tag{1}$$

The goal is to minimize the maximum load on any server, ensuring an even distribution task. Static LB algorithm and dynamic LB algorithm shown in *Figs. 2* and *3*.

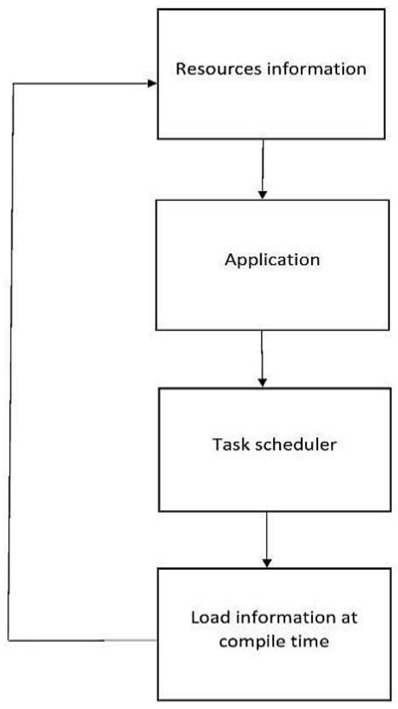

Fig. 2. Static LB algorithm.

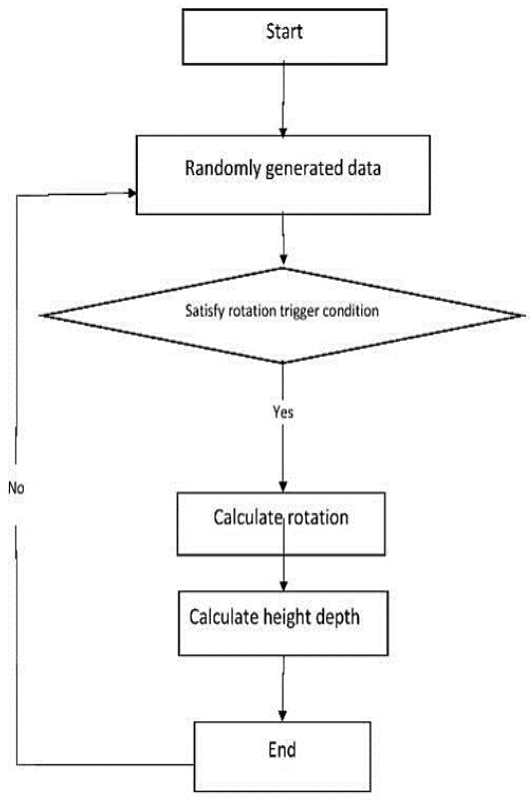

Fig. 3. Dynamic LB algorithm.

II. Task distribution

$L_i = \sum_{j=1}^{n} Task_j / Si.$

Where L_i is the load on server i, $Task_j$ represents individual tasks, and S_i i the capacity of server i.

These are conceptual equations that summarize the main ideas in the document. They will form the background by which the different load-balancing algorithms will actually attempt to balance work practically by pointing out, in effect, a valid division of work across available servers in a cloud computing environment [5]. *Fig. 4* illustrates the key challenges associated with load balancing in cloud computing environments. These challenges include:

I. Distributed geographical nodes: Managing nodes that are spread across various locations.

II. Single point of failure: Preventing system breakdown due to reliance on a single component.

III. Virtual machine migration: Efficiently moving virtual machines across different hosts.

IV. Heterogeneous nodes: Handling differences in hardware and software configurations among nodes.

V. Scalability of load balancer: Ensuring the load balancer can scale to accommodate growing workloads.

VI. Complexity of algorithm: Developing and implementing efficient load balancing algorithms.

VII. Automated service provisioning: Automatically allocating resources based on demand.

VIII. Energy management: Optimizing energy consumption during load balancing processes.

These challenges highlight the complexities involved in ensuring effective load balancing for cloud computing infrastructures.

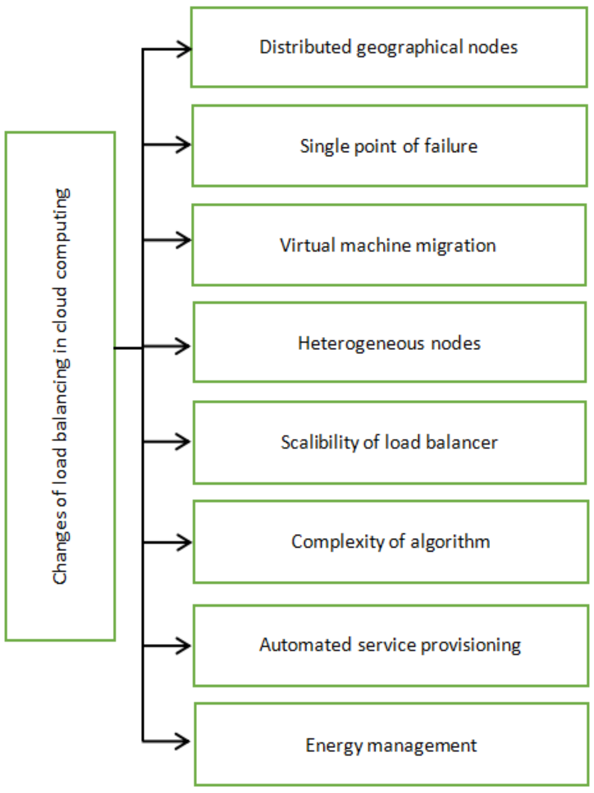
Changes of load balancing in cloud computing
Distributed geographical nodes
Single point of failure
Virtual machine migration
Heterogeneous nodes
Scalibility of load balancer
Complexity of algorithm
Automated service provisioning
Energy management

Fig. 4. Taxonomy of critical LB.

Table 1. History of algorithm of LB.

Algorithm Name	Year Proposed/ Developed	Type of Algorithm	Objective	Input Parameters	Routing Strategy	Adaptability	Use Cases
Shortest job first	1950s	Centralized, static	Minimize response time	Job duration, queve length	Centralized	No	Batch processing system, task schedulin gin operating system
Min max	$2010s$	Centralized, dynamic	Optimized resource utilization	Resource demand, server capacity	Centralized	Yes	Cloud computin g environment, resource provisioning in distribute d system
Weighted Round Robin	1969	CPU Scheduling (preemptive)	Mechanism for assigning different weight to different task of processes	Server, weight	Distributing Incoming request or traffic among many servers based on their weight	Ability to dynamic ally adjust to change in conditio n in network or server environ ment	Web server LB, Application server LB
Dynamic weighted round robin	2005	Dynamic, Weighted	Maximize throughput	Server load, server capacity, request rate	Weighted round robin	Yes	Web server, cloud Computing g environment

1.2| History of Load Balancing Algorithms

In Early 2000s, LB is done using simple round - robin algorithm, where tasks or requests were distributed equally to available servers. There was huge usage of simple static algorithms because of the factors like server capacity or response time [6]. This method actually doesn't focus on actual load on individual servers and the type of task they are performing. This method was better for less complex server's environment. The incoming Request were handled with some predefined protocols without any real-time adjustments. In Mid 2000s, virtualization technologies were started to gain more prominence, which leads for the creation of Virtual Machines (VMs). Which gave us the simulated computing environments running on a physical host machine [7]. Virtualization provided a more flexible and dynamic way to allocate computing resources a bit more efficiently. Due to the virtualization, the traditional LB methods became less effective, the Dynamic nature VMs workloads were not constant, and the resource needs are changing rapidly, to resolve that issue Dynamic LB algorithms were introduced. These algorithms monitor and handle the load in real time on servers and adjust their distribution of incoming tasks accordingly [8].

In Late 2000s, there we couple of could service providers like Amazon web Services, google cloud, and Microsoft Azure were started themselves establishing. A new architecture was established called distributed architecture in which applications and services could able to run on servers located across multiple data centers or regions. The distributed nature created a new challenge for LB. Some systems introduced centralized controllers also known as load balancers. These controllers consider factors like overall server load, network conditions, and the availability of resources across different data centers. This help in making more globally optimized LB decisions [9].

In 2010s, as cloud environments became more complex with various type of applications and services. Content-based routing gained prominence for optimize resource allocation based on the type of content or service being delivered [10]. By tailoring resource allocation to the unique needs of each application, application-ware and content-based approaches contribute to enhanced performance. LB algorithms need to dynamically adjust to fluctuations in application demands, requiring real-time monitoring and adaptive strategies. In 2015 onwards, due to the complex designs and environments and the variability of workloads, LB algorithms started using machine learning techniques for dynamic adaption [11]. Machine learning helps load balancers to learn from historical data and current patterns in given tasks. This helps them to make more precise decisions about distributing the resources. Machine learning in LB might further integrate with other technologies such as edge computing and advanced analytics, offering more comprehensive solutions for optimizing resource usage [12].

In 2020s, Organizations increasingly adopted hybrid cloud architectures, combining public cloud services with on-premises infrastructure. LB algorithms evolved to seamlessly distributed workload across both cloud environments and traditional data centers. Hybrid and multi-cloud environments bring challenges related to interoperability. LB algorithms need to be capable of seamlessly managing and distributing workloads across diverse infrastructure while considering factors such as latency and data transfer costs [13]. The solution to enhance flexibility, organizations sought LB solutions that are vendor-agnostic, allowing them to easily transition workloads between different cloud providers. Introduction of edge computing helps to reduce the latency by process the data closer to the source of generation. Edge computing also helps in improving responsiveness by reducing the distance data needs to travel for processing. As edge computing evolves, LB algorithms are likely to become more intelligent at the edge, incorporating machine learning and predictive analytics to optimize resource usage based on real-time conditions [14].

2|Literature Review

LB involves distributing workloads across multiple servers, enhancing throughput, and minimizing response times within a cloud environment. This allocation ensures that incoming workloads are divided among available servers and computing resources, optimizing resource utilization and response times. Load balancers employ various algorithms to determine the server to which incoming requests should be directed [15]. LB strategies are typically categorized as static or dynamic. Initially, static LB was the prevalent approach [16]. In the realm of cloud-based LB algorithms, two primary types are commonly employed: static algorithms such as round-robin and randomized algorithms, and dynamic algorithms such as throttled and active VM monitoring. Static algorithms allocate loads to VMs without considering the VMs' states beforehand. Their primary objective is to minimize response times; thus, they overlook parameters such as load status. In LB scenarios within cloud environments, the optimal scenario arises when the initial or randomly selected VM is available or idle for task assignment [17].

Shortest Job First (SJF) prioritizes the execution of waiting processes with the smallest execution time. While it minimizes CPU cycles and execution time, it may lead to starvation for newer, shorter processes. SJF is beneficial for resource utilization and avoiding starvation, but it's not suitable for heterogeneous resources and its performance depends on the number of tasks it handles [18]. Round Robin Algorithm distributes requests sequentially, utilizing a round-robin approach to assign jobs. It randomly selects the first node and then assigns tasks to other nodes in a circular manner. While it facilitates faster response times for similar workload distributions, it may not optimize resource utilization, potentially leading to idle resources and degraded performance [19]. Weighted Round Robin Algorithm designed with prescribed weights; this algorithm assigns jobs based on these weight values. Processors with higher abilities receive larger weight values, ensuring that higher-weighted servers handle more tasks. This approach maintains steady traffic distribution when all weights are at a similar level [20].

Min to Max Algorithm identifies tasks with minimum completion times and selects the one with the maximum value from this subset. Tasks are then scheduled on the machine according to this maximum time period. While it focuses on load distribution and resource allocation, it may not effectively address the performance of heterogeneous resources and tasks. Min to Max Algorithm (Variation) identifies tasks with minimal completion times [21]. It then selects the task with the maximum value from this subset and allocates it to the node. While it emphasizes resource allocation and distribution, it may not specifically target overall system performance improvement [22]. *Fig 2* depicts the static LB algorithm.

The dynamic LB algorithm allocates the jobs to the node only at run time. In this algorithm the jobs can be reassigned depending on the situation at the runtime. The jobs will be migrated from over utilized node to the underutilized node [23]. A LB algorithm inspired by Honey Bee Behavior was proposed, with the goal of achieving balanced loads across VMs to optimize throughput and prioritize task distribution on the VMs. The primary objective is to minimize the waiting time of tasks in the queue. Throttled LB algorithms operates by maintaining an indexing table that tracks the current load of all VMs. Upon receiving an incoming task, the algorithm scans this table to identify an under-loaded VM and allocates the task accordingly. If the VM is available, the task is assigned to it. The indexing table is continuously updated during task allocation and deallocation processes. This method effectively minimizes waiting times and enhances resource utilization by evenly distributing the load across VMs. However, a drawback of this approach is its susceptibility to a single point of failure. As the number of tasks increases, it can lead to performance degradation within the system. *Fig 3* depicts the dynamic LB.

3 | Proposed Study

In today's computing world, LB has evolved from simply distributing work across multiple servers to a more complex concept that aligns with the needs of power and distribution. As cloud computing, the Internet of Things (IoT), and edge computing grow, the traditional idea of fixed and predictable workplaces is being challenged. LB algorithms designed for these environments often struggle to adapt to the variability and uncertainty inherent in these new situations. Therefore, there is a pressing need for new methods that can quickly adapt to the changing nature of work. Additionally, distributing resources in the edge computing environment poses unique challenges such as latency, bandwidth limitations, and communication overhead, requiring a balance that can effectively divide tasks based on the capacity and limitations of the edge. This discussion aims to describe and address these efforts, paving the way for the development of more effective LB solutions.

3.1 | Rresearch Gap

Existing studies often assume a relatively stable and predictable workload environment. However, as dynamic and unpredictable workloads increase in cloud computing, IoT, and edge computing, LB algorithms that can adapt to changing workloads in real-time are needed. In environments with dynamic workloads, the challenge is to develop LB algorithms that can dynamically adapt to the changing requirements of modern computing systems. Existing algorithms often assume a certain level of predictability that may not apply in scenarios such as cloud computing or the Internet of Things (IoT). Research in this area should focus on creating adaptive algorithms that can make real-time adjustments based on factors such as workload changes, resource availability, and system responsiveness. This includes exploring new heuristics, machine learning models, or hybrid approaches that can effectively manage the uncertainty associated with dynamic workloads [7].

LB in edge computing environments poses unique challenges due to the distributed nature of resources and limitations of edge devices. There is a research gap in developing algorithms that address latency, bandwidth limitations, and communication overhead in edge computing scenarios. Edge computing poses unique challenges due to the distributed nature of resources, limited bandwidth, and the need for low-latency responses. LB at the edge requires algorithms that can efficiently distribute tasks based on the limitations and capabilities of edge devices. Research should explore LB strategies that optimize low-latency communications, account for the variability of network conditions, and dynamically adapt to the availability of edge resources. Additionally, research on edge computing-aware scheduling algorithms that consider the geographic distribution of tasks and data can contribute to more efficient LB in edge computing scenarios [8].

3.2 | Finding

This segment delineates the proposed framework designed to enhance LB in a Cloud Computing Environment. The primary objective of this framework is to establish a highly available cloud environment, mitigating system failures and facilitating the recovery of user tasks. Ultimately, this improves security in Cloud Computing applications [11]. In the dynamic landscape of cloud computing, LB emerges as a pervasive issue impacting network performance. Previous research has introduced numerous techniques to address LB in the network. However, existing methods often necessitate additional hardware and software, thereby escalating system complexity. There is a pressing need for a technique that circumvents the requirement for extra hardware and software to manage network load. Hence, a novel approach is warranted that streamlines the process of LB without imposing supplementary steps or dependencies [9]. Cloud computing confronts numerous challenges, with LB emerging as one of the most critical issues requiring specific attention. This encompasses challenges such as VMs migration, ensuring VMs security, prioritizing user Quality of Service (QoS) comfort, and optimizing resource utilization. These challenges are given equal consideration in the pursuit of enhancing cloud resource management. Presented below is a selection of key LB issues, while Figure 4 illustrates the taxonomy of critical LB concerns [10].

Cloud data centers are often distributed across various locations to efficiently process customer requests. However, existing LB approaches in literature overlook factors like network and communication delays, spatial constraints within computing nodes, and customer resource availability. Algorithms designed for such distributed environments may not be suitable for nodes in remote areas Certain LB algorithms proposed in literature rely on centralized decision-making, posing a risk of system failure if the central node malfunctions. This centralized approach can undermine the overall reliability of the computing system [10]. Virtualization enables multiple VMs to operate on a single physical device, each with distinct settings. When a physical device becomes overloaded, LB methods facilitate the migration of VMs to remote locations to alleviate the strain on the system [11]. Initial inquiries into cloud LB primarily focused on homogeneous nodes. However, cloud consumers require a dynamic solution that can efficiently operate on heterogeneous nodes to optimize network performance and reduce response times [10].

Traditional storage devices in cloud computing incurred significant resource and equipment costs. Cloud services offer consumers the ability to store data heterogeneously, eliminating control issues. However, managing data storage involves duplicating stored data for improved accessibility and data continuity [10]. Cloud services offer accessibility and on-demand scalability, allowing users to scale resources up or down as needed. Effective LB should accommodate rapidly changing computational requirements, memory usage, and device topology to ensure optimal performance. Cloud computing algorithms must strike a balance between speed and simplicity to optimize system efficiency and quality. Complex algorithms can hinder cloud performance and degrade system quality.

A key feature of cloud computing is its flexibility in resource provisioning. However, ensuring the efficient use of cloud services while maintaining productivity comparable to conventional systems remains a challenge. Effective strategies are needed to manage resource allocation and utilization in the cloud environment [10]. LB algorithms are classified based on several criteria, following a top-down approach in the classification process. Existing review papers often lack a comprehensive hierarchical taxonomical classification of LB algorithms, making it challenging to determine where a specific algorithm fits within the taxonomy. The classification criteria include the nature of algorithm, state of algorithm, trait used for LB, type of LB, and technique used in LB. 'For the first time in literature, this work provides an in-depth analysis of LB algorithms, addressing the deficiencies observed in previous studies. Based on the nature of the algorithm, LB algorithms are categorized as either proactive or reactive. Regarding the state of the system, LB algorithms are further classified as static, dynamic, or hybrid. On the basis of the trait used in LB, algorithms are categorized as scheduling and allocation algorithms. In terms of the type of LB, algorithms are grouped as VM LB algorithms, CPU LB algorithms, Task LB algorithms, Server LB algorithms, Network LB algorithms, and Normal Cloud LB algorithms. Regarding functionality, LB methods are grouped as hardware LB, elastic LB, with the latter further divided into network LB, application LB, and classic LB. Finally, based on the technique employed, LB algorithms are classified into Machine learning, Evolutionary, Nature-Inspired, Mathematical-derived algorithms, and Swarm-based techniques [12].

Table 2. Configurational analysis.

Configuration Type	Property
Hardware configuration	We utilized servers with Intel Xeon processors, ranging from quad-core to Octa-core configurations, accompanied by 32GB to 128GB of RAM and SSD storage ranging from 500GB to 1TB. Additionally, considerations were made to ensure scalability and capacity to accommodate varying loads.
Software configuration	The software stack comprised robust operating systems, primarily Linux distributions tailored for server environments. We deployed Ubuntu Server 20.04 LTS and CentOS 8.3 across our server infrastructure. LB algorithms were implemented using custom scripts developed in Python, leveraging libraries such as NumPy and SciPy for computational tasks.
Network topology	Our network infrastructure featured a hierarchical topology with redundant links to ensure fault tolerance and high availability. Servers were interconnected via gigabit Ethernet switches, facilitating low-latency communication.
Workload characteristics	The workload simulated in our experiments encompassed a variety of realistic scenarios encountered in modern computing environments. This included a mix of web requests, database queries, and other application-specific tasks, with varying levels of complexity and resource requirements. Request arrival rates were modeled to reflect dynamic fluctuations in demand, with peak periods and off-peak intervals accounted for.
Experimental setup	Our experimental methodology followed a systematic approach aimed at comprehensively evaluating LB algorithms. We systematically varied parameters such as server capacity, workload intensity, and algorithm configurations to assess their impact on performance.
Metrics and measurement tools	Performance evaluation relied on a set of carefully chosen metrics, including throughput, response time, and server utilization.
Data collection	Data collection during experiments was facilitated through comprehensive logging mechanisms deployed across the infrastructure. System metrics such as CPU utilization, memory usage, and network throughput were logged at regular intervals.
Experimental constraints	Despite meticulous planning, our experimental setup was subject to certain constraints that may have influenced the scope and interpretation of results.

3.3 | Analysis

The historical evolution of LB algorithms reveals a progression from rudimentary approaches in traditional computing environments to sophisticated techniques tailored for modern distributed systems and cloud infrastructures [13]. Our analysis traces the development of LB strategies through key milestones and innovations, shedding light on the factors driving their evolution and the implications for contemporary computing architectures. Early LB techniques, such as Round Robin and Least Connections, provided rudimentary mechanisms for distributing workloads across multiple servers. While these approaches helped alleviate the burden on individual servers, they lacked adaptability and struggled to cope with dynamic workloads and heterogeneous resource environments [15]. The emergence of distributed systems in the late

20th century spurred advancements in LB algorithms to address the challenges posed by decentralized architectures. Techniques such as Weighted Round Robin and Least Loaded emerged to optimize resource utilization and improve performance in distributed computing environments. These algorithms introduced the concept of assigning weights or priorities to servers based on their capacity and workload, allowing for more efficient load distribution.

The evolution of LB algorithms accelerated with the rise of cloud computing, where dynamic resource provisioning and elastic scalability became essential requirements. Dynamic LB techniques, including Dynamic Weighted Round Robin and Dynamic Least Connections, emerged to adaptively allocate resources based on real-time workload conditions [14]. These algorithms leverage dynamic monitoring and feedback mechanisms to adjust load distribution dynamically, ensuring optimal performance and resource utilization in fluctuating environments. Machine learning and Artificial Intelligence (AI) have also made significant contributions to the evolution of LB algorithms. Research efforts in this area have explored the application of predictive analytics and optimization techniques to improve LB decisions. Machine learning-based approaches offer the potential for more intelligent and adaptive LB strategies capable of learning from past behavior and predicting future workload patterns [15].

3.4 | Discussion

The historical evolution of LB algorithms reflects a continuous quest for more efficient, adaptive, and scalable strategies to manage resource allocation in distributed computing environments. Our analysis underscores several key insights and implications for both research and practice in the field of LB: Firstly, the evolution of LB algorithms has been driven by the increasing complexity and scale of distributed systems [16]. As computing architectures have become more decentralized and dynamic, the need for sophisticated LB techniques capable of handling heterogeneous resources and fluctuating workloads has become paramount. Secondly, the integration of machine learning and AI into LB algorithms represents a promising frontier for future research [15]. By harnessing the power of data-driven analytics and predictive modeling, machine learning-based approaches offer the potential to optimize LB decisions in real-time and adapt to evolving workload patterns with greater accuracy.

Furthermore, the historical evolution of LB algorithms highlights the importance of considering practical constraints and trade-offs in algorithm design. While more complex algorithms may offer superior performance in certain scenarios, they may also incur higher computational overhead or require more extensive configuration and tuning. Looking ahead, future research directions in LB algorithms may focus on addressing emerging challenges such as edge computing, containerization, and hybrid cloud architectures [17]. By exploring innovative approaches to LB that take into account the unique characteristics of these environments, researchers can continue to advance the state-of-the-art in distributed computing and cloud infrastructure management. In conclusion, the historical evolution of LB algorithms serves as a testament to the ongoing pursuit of efficiency and optimization in distributed computing environments. By understanding the lessons learned from past developments and embracing emerging technologies and methodologies, researchers and practitioners can drive further innovation in LB and contribute to the continued advancement of cloud computing and beyond [15].

4 | Limitation and Future Scope

The limitations of current measurement systems arise from many factors, including scalability challenges, complexity, and the need to adapt to a dynamic network environment. While the integration of machine learning and AI technology holds promise for improving employment stability, there are also important limitations to consider [18]. A major limitation is that large amounts of performance data are needed to train machine learning models for predictive modeling. Obtaining and managing such information can be potentially abusive and raise privacy concerns, especially in sensitive environments [20]. Additionally, machine learning models can have common problems with network opacity or suffer from biases if not carefully designed and trained. Also, incorporating AI into the process equation makes calculation more expensive and difficult. Real-time decision-making in a networked environment requires efficient processes that can be adapted quickly; This can create difficulties in implementing a process or an environment with limited activity. Interdisciplinary research is essential to address these limitations and guide future progress in a balanced way [21]. Collaboration between computing, communications, intelligence, and cybersecurity experts can help develop more robust, scalable, and secure LB solutions.

The future trajectory of LB algorithms presents a rich landscape of opportunities and challenges at the intersection of computing, networking, and AI. One prominent avenue for further exploration is the integration of Machine Learning (ML) and AI techniques into LB mechanisms. By leveraging ML models for predictive analytics and AI-driven decision-making processes, load balancers can dynamically adapt to changing network conditions and application demands [1]. As computing paradigms evolve, the advent of edge computing and Internet of Things (IoT) architectures necessitates LB strategies tailored for distributed edge nodes and heterogeneous device ecosystems. Research in this area focuses on developing lightweight, context-aware algorithms capable of optimizing resource allocation and minimizing latency in edge environments [2].

The ongoing proliferation of hybrid and multi-cloud infrastructures underscores the importance of seamless workload distribution and resource management across diverse cloud platforms. Future LB algorithms are expected to incorporate federated learning principles and dynamic provisioning techniques to ensure efficient utilization of resources while maintaining high availability and performance [3]. Security remains a paramount concern in modern computing environments, leading to the exploration of security-aware LB strategies. These strategies integrate threat intelligence, encryption protocols, and access control mechanisms into LB decisions to mitigate cybersecurity risks such as DDoS attacks and data breaches [4]. Quantum computing represents a paradigm shift in computational capabilities, prompting researchers to investigate quantum-inspired LB algorithms optimized for quantum computing architectures. This emerging field aims to harness quantum principles such as superposition and entanglement to achieve unprecedented levels of scalability and efficiency in load distribution [5]. Ethical considerations also shape the future development of LB algorithms, emphasizing principles of fairness, transparency, and environmental sustainability. LB frameworks incorporating ethical guidelines ensure equitable resource allocation, transparent decision-making processes, and energy-efficient resource utilization practices [6].

5 | Conclusion

The evolution of LB algorithms in the context of cloud computing and distributed systems has witnessed significant advancements. This research paper provided a comprehensive overview of the historical progression of LB algorithms, from rudimentary algorithms such as FCFS and Round Robin to dynamic and heterogeneous resource environment. The challenges of existing LB algorithms are critically evaluated, highlighting the need for adaptive and real-time LB solutions. The integration of machine learning and AI into LB can be helpful. By using the power of data-driven analytics and machine learning-based approaches offer the potential to optimize LB decisions in real-time. The historical evolution of LB algorithms serves as a testament to the ongoing pursuit of efficiency and optimization in distributed computing environments. By understanding the lessons learned from past developments and embracing emerging technologies and methodologies, researchers and practitioners can drive further innovation in LB [4].

Acknowledgments

We would like to thank DR. Hitesh Mohapatra provided valuable guidance and support throughout the study. We also thank the Department of Computer Science and Engineering for providing the necessary resources and facilities. We thank our peers for their positive comments and support. Finally, we would like to thank our family and friends for their support and always encouraging us.

Author Contribution

Conceptualization, C.T. and D.P.; Methodology, D.P.; Validation, I.C., C.T., and D.P.; Formal analysis, A.P.; Investigation, C.T.; Resources, D.P.; Data curation, C.T.; Writing—original draft, I.C.; Writing—review & editing, I.C. ; Visualization, I.C.; Supervision, A.P. ; Project administration, A.P.; All authors have read and agreed to the published version of the manuscript.

Data Availability

In the course of our research, we utilized preexisting data sets sourced from previously published studies. The data sets employed in our analysis were obtained from the preexisting research papers listed in the reference section below, which provided a comprehensive repository of data relevant to our study's objectives. Our use of these data sets allowed us to build upon existing knowledge and validate our hypotheses within a well-established framework.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] Youssf, O., ElGawady, M. A., & Mills, J. E. (2016). Static cyclic behaviour of FRP-confined crumb rubber concrete columns. *Engineering structures*, *113*, 371–387. https://www.sciencedirect.com/science/article/pii/S0141029616000493

[2] Vijay, R., & Sree, T. R. (2023). Resource scheduling and load balancing algorithms in cloud computing. *Procedia computer science*, *230*, 326–336. https://www.sciencedirect.com/science/article/pii/S1877050923020938

[3] Kaviarasan, R., Balamurugan, G., Kalaiyarasan, R., & Venkata RavindraReddy, Y. (2023). Effective load balancing approach in cloud computing using inspired lion optimization algorithm. *E-prime - advances in electrical engineering, electronics and energy*, *6*, 100326. https://www.sciencedirect.com/science/article/pii/S2772671123002218

[4] Devi, D. C., & Uthariaraj, V. R. (2016). Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks. *The scientific world journal*, *2016*(1), 3896065.

[5] Mohapatra, H., & Rath, A. K. (2020). Fault-tolerant mechanism for wireless sensor network. *IET wireless sensor systems*, *10*(1), 23–30. https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-wss.2019.0106

[6] Sarkar, F., & Das, S. K. (1997). Design and implementation of dynamic load balancing algorithms for rollback reduction in optimistic pdes. *Proceedings fifth international symposium on modeling, analysis, and simulation of computer and telecommunication systems* (pp. 26–31). IEEE.

[7] Mekonnen, D., Megersa, A., Sharma, R. K., & Sharma, D. P. (2022). Designing a component-based throttled load balancing algorithm for cloud data centers. *Mathematical problems in engineering*, *2022*(1), 4640443.

[8] Mohapatra, H., & Rath, A. K. (2019). Fault tolerance in WSN through PE-LEACH protocol. *IET wireless sensor systems*, *9*(6), 358–365. https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-wss.2018.5229

[9] 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

[10] Mallikarjuna, B., & Reddy, D. A. K. (2019). The role of load balancing algorithms in next generation of cloud computing. *Control syst*, *11*, 20.

[11] Mohapatra, H., & Rath, A. K. (2019). Detection and avoidance of water loss through municipality taps in India by using smart taps and ICT. *IET wireless sensor systems*, *9*(6), 447–457. https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-wss.2019.0081

[12] Dhari, A., & Arif, K. I. (2017). An efficient load balancing scheme for cloud computing. *Indian journal of science and technology*, *10*(11), 1–8.

[13] Kalpana, M. S. (2019). Load balancing in cloud computing with enhanced genetic algorithm. *International journal of recent technology and engineering*, *8*(2S6).

[14] Shahid, M. A., Islam, N., Alam, M. M., Su'ud, M. M., & Musa, S. (2020). A comprehensive study of load balancing approaches in the cloud computing environment and a novel fault tolerance approach. *IEEE access*, *8*, 130500–130526. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9139971

[15] Mohapatra, H., & Rath, A. K. (2020). Survey on fault tolerance-based clustering evolution in WSN. *IET networks*, *9*(4), 145–155. https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-net.2019.0155

[16] Shafiq, D. A., Jhanjhi, N. Z., & Abdullah, A. (2022). Load balancing techniques in cloud computing environment: a review. *Journal of king saud university - computer and information sciences*, *34*(7), 3910–3933. https://www.sciencedirect.com/science/article/pii/S131915782100046X

[17] Afzal, S., & Kavitha, G. (2019). Load balancing in cloud computing–A hierarchical taxonomical classification. *Journal of cloud computing*, *8*(1), 1–24.

[18] Mohapatra, H., & Rath, A. K. (2021). Fault tolerance in WSN through uniform load distribution function. *International journal of sensors wireless communications and control*, *11*(4), 385–394. https://scholar.google.com/citations?user=hZquqF8AAAAJ&hl=en&oi=sra

[19] Gajbhiye, A., & Singh, D. S. (2017). Global Server load balancing with networked load balancers for geographically distributed cloud data-centres. *International journal of computer science and network*, *6*(6), 682–688.

[20] Sajjan, R. S., & Yashwantrao, B. R. (2017). Load balancing and its algorithms in cloud computing: a survey. *International journal of computer sciences and engineering*, *5*(1), 95–100.

[21] Chandra, H., & Bahuguna, H. (2017). A survey of load balancing algorithms in cloud computing. *International journal of computer engineering and applications*, *11*(12).

[22] Ramathilagam, A., & Vijayalakshmi, K. (2016). A survey of scheduling algorithm in cloud computing environment. *International journal of control theory and applications*, *9*(36), 137–145.

[23] Sharma, H., & Semwal, P. (2021). A review of load balancing algorithms in cloud computing. *International journal of creative research thoughts*, *9*(3), 2786–2791.