Smart Internet of Things

www.siot.reapress.com

Smart. Internet. Things. Vol. 1, No. 1 (2024) 67–82.

Paper Type: Original Article

From Theory to Practice: Addressing Load Balancing Challenges in Real-Time Cloud Applications

Vaidehi Gupta[1],*, Vaibhav Sahay[1]

[1] Department of Computer Engineering, KIIT University, Bhubaneswar, India; vaidehi14nv@gmail.com; vsahay717@gmail.com.

Citation:

Received: 06 March 2024 *Revised: 11 June 2024* *Accepted: 16 August 2024*	Gupta, V., & Sahay, V. (2024). From theory to practice: addressing load balancing challenges in real-time cloud applications. *Smart Internet of Things*, *1* (1), 67-82.

Abstract

The world is currently steered by the paradigms of cloud technology. The internet tools work in alignment with cloud technologies, since they have become its essential components. It addresses challenges like load balancing which ensures efficient distribution of traffic among multiple servers for optimal service delivery. This paper is therefore dedicated to finding out how to identify and solve specific challenges related to load balancing in real time cloud computing applications. The existing study therefore puts in doubt the notion that universal load balancing technology can be used for all cloud applications. This revelation highlights a notable void in literature and practices, expressing the need for customised strategies responsive to customers' needs on a real time basis. Our results show that if adapted balancing solutions are used, instead of traditional techniques, real-time system's performance can profit from improved efficiency and reliability. Hence, this work enhances our understanding of the subtleties of cloud computing, especially those issues which affect its ability to perform well in diverse applications. These findings pave the way for future research on more reliable and efficient cloud services that can increase overall impact on technological growth as well as influence both digital and physical environments.

Keywords: Cloud computing, Load balancing, Load balancing algorithms, Real-time applications.

1 | Introduction

The birth of the cloud computing era was ushered in by the development of computing technologies, which in turn brought about a new paradigm for user services by enabling pay-per-use access to IT services from any location at any time. Many businesses are moving their operations to the cloud due to the flexibility of cloud services, and service providers are expanding their data centers to accommodate more users [1]. The difficulties with stream computing get worse as enterprise clusters get bigger and store more data. Therefore, real-time handling of dynamic and volatile data streams is necessary [2].

Corresponding Author: vaidehi14nv@gmail.com

Future IT real-time applications will keep pushing for performance scaling in everything from servers to sensors. Real-time applications are anticipated to offer a high degree of parallelism, where real-time operations can employ many cores simultaneously in order to fully take advantage of multicore CPUs. A scheduling strategy that provides high scheduling and efficient load balancing is necessary to provide real-time performance while optimising multicore resources [3].

The variability of demand in a cluster system makes it more challenging to distribute the load evenly among its nodes. Furthermore, a large degree of fluctuation generally results in incorrect load balancing decisions being made with outdated information, which makes it challenging to make corrections in real time while an application is running [4]. Data centers can prevent Virtual Machine (VM) overloading and/or under-loading by using load balancing in the cloud. This is a challenge in the realm of cloud computing.

Consequently, developers and researchers must create and execute a suitable load balancer for distributed and parallel cloud settings [5]. The main purpose of Load Balancing Algorithms (LBAs) is to minimise the overall job execution time while optimising processor utility and distributing the load evenly across them. This suggests that there should be as little variation as possible between processors that are the most and least loaded. Consequently, in order to improve the effectiveness of the load-balancing mechanism for real-time applications of the aforementioned paradigm, the load information on each processor must be updated [4].

The load balancer's primary goal is to distribute resources evenly among tasks in order to maximise resource efficiency, user satisfaction, and minimum cost. It also produces high-quality work, gripping rapid traffic blasts that sustain website traffic and elasticity, all of which encourage us to identify and address LB issues as they arise [6]. This is essential to guaranteeing that clients, associates, and end users of cloud-based apps may access them with ease. The potential of load balancing and its many uses serve as inspiration for discussing this significant task, identifying the main problems with LB, and finding solutions for them [7].

By doing calculations closer to the user or data source, the main objective is to minimise latency and bandwidth utilisation and do away with the necessity of sending massive volumes of data to a centralised server. Additionally, it encourages serverless design, which makes application scaling and deployment simpler. This lowers operating costs and boosts efficiency [8].

With the growing number of multicore processors, load balancing for workloads has recently received renewed attention. To ensure that each core is working at the same pace, load balancing aims to spread the computational load equally. It is possible to use resources effectively without over provisioning or wasting them by distributing the workload between cores.

Furthermore, combining balancing with the DVS approach can greatly minimise energy use, as demonstrated in. We anticipate that balanced loads can also result in other important advantages, including increased dependability, phase-change memory-related wear management, and throughput optimization [9].

There are two primary types of load-balancing techniques: dynamic and static. Static approaches disregard the current system load in favour of preset timetables based on projected demand. This could result in ineffective task allocation since it necessitates collecting static information about every node in advance. On the other hand, dynamic load balancing dynamically modifies the distribution of tasks among processing units by continuously analysing the system load in real time. For best results, this strategy seeks to balance each processor's workload according to its capabilities [10].

Fig. 1 represents the difference between the Static v/s Dynamic Load Balancing in Cloud Computing.

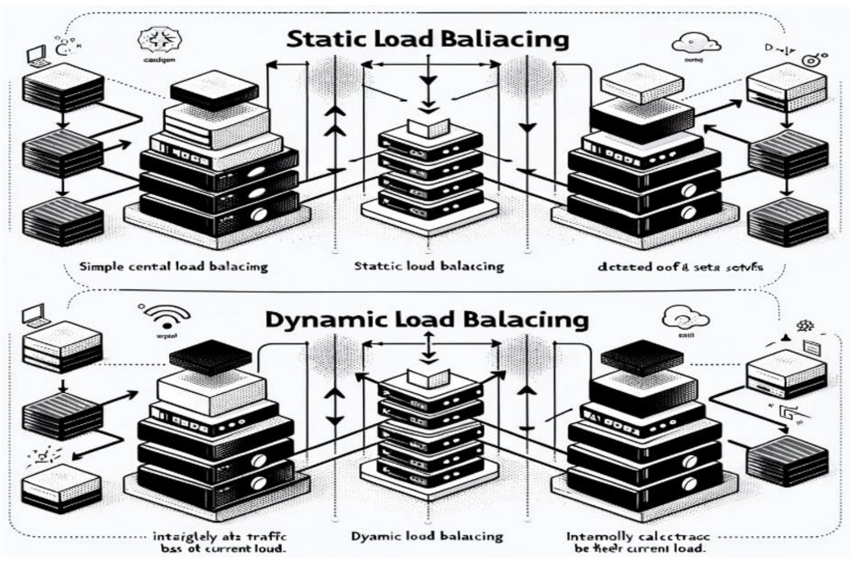
Static Load Baliacing
Simple cental load balacing
Statcic loud balatcing
dctetsd oof d seta sctvfs
Dynamic Load Balacing
intaiglely ats traffc bas ot current loud.
Dyamic lood balatcing
Intamolly calcctracc be fieer current load.

Fig. 1. Difference between static v/s dynamic load in cloud computing.

The project's goal is to create and put into practice a distributed process scheduling mechanism for the hierarchical cluster-based distributed real-time system model that was previously designed. In this paradigm, the system is made up of groups of computer nodes. Every cluster has a synchronous ring protocol in operation. Representatives from each cluster oversee the clusters. Additionally, there is a higher-level ring made up of the lower-level cluster representatives. The hierarchy may consist of two or more levels. A leader oversees the entire system at the highest level. A distributed real-time platform for distributed real-time is what the system is intended to offer [11].

A group of either heterogeneous or homogeneous systems is referred to as a parallel computing system. Parallel computing system's program execution may use varying numbers of processors at different times within an instruction cycle. Parallel processing is a recently developed idea that can run many jobs on separate processors at the same time. It is a useful method for resolving difficult problems requiring a lot of computation [12]. Depending on the type of processor, a parallel system can be classified as either homogeneous or heterogeneous. A multiprocessor system is the fundamental element of a parallel system (MTS). Multiple processing elements, multiple memory modules, and numerous input/output modules are found in MTSs. Each CPU has the entitlement to utilise any memory modules and any input-output components. The Multiprocessor Interconnection Network (MIN) handles the connections between them [4]. Numerous processing components, referred to as nodes, make up the MTS. These systems are calculated in terms of low degree, reduced diameter, complexity, and cost, among other parametric topological qualities. Many multiprocessor systems, such the Folded Crossed Cube, Hypercube, and Crossed Cube, have been proposed [13]. Hypercube (HC), however, is a typical MIN. An HC network consists of n nodes with n linkages between each node. Despite its commercial acceptance, the hypercube interconnection network—which has the potential to handle data in parallel—has an exponential 2-fold extension. The HC network's debruijn network has the primary example [14].

2|Literature Review

2.1|Cloud Computing Environment and Big Data effects

This study aims to explore the challenges and opportunities of load balancing in the context of cloud computing in real-time applications. Cloud computing, a relatively new technology, mostly refers to a centralised storage area made possible by distant machines running internet-based software [15]. To increase productivity, users will disperse resources and services around the network. There are several methods for choosing and assessing cloud computing [16]. The number of Internet users and Internet of Things (IoT) devices is rising in this Big Data era, and so is the volume of data that needs to be processed in real time. The difficulties with cloud computing get worse as enterprise clusters get bigger and store more data [2]. Users can access a large pool of shared resources, such as software, data, storage, and applications, whenever they need them with cloud computing. Due to the technology's rapid development, a sizable user base demanding higher levels of customer satisfaction has been drawn in. In this quickly changing environment, load balancing has become an important field of study [17]. This study addresses the shortcomings of the real-time data stream collection system Apache Flume by suggesting enhancements to load balancing and storage. To cut down on disk and network overhead, they provide a memory-file channel storage technique and a load balancing strategy based on free memory. Experiments conducted under a variety of situations, including a bad network, high availability demands, memory resource rivalry, and big data sizes, demonstrate considerable gains in availability (above 99.999%) and performance (10–50%) [14]. *Fig. 2* represents Cloud Computing and Big Data as a perfect combination [18].

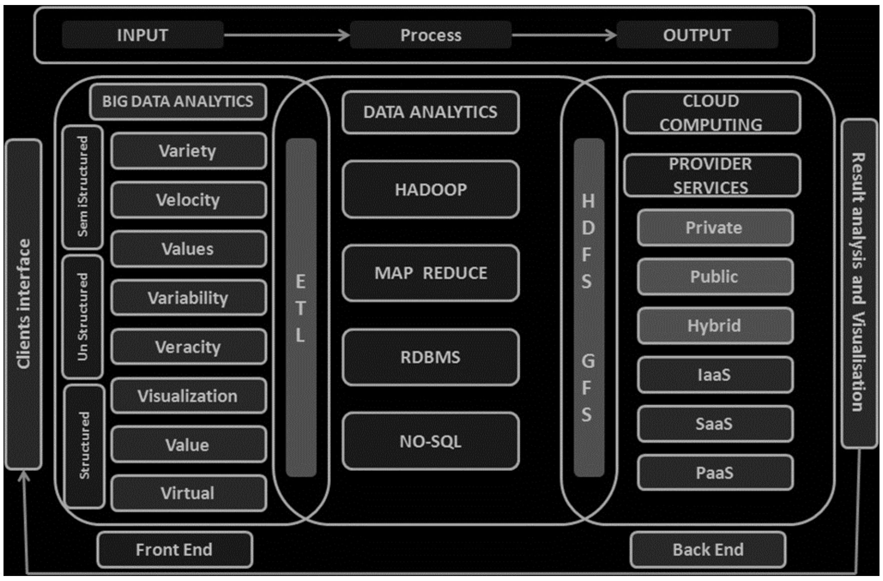

Fig. 2. Cloud computing and big data-a perfect combination.

2.2 | Requirement of an Efficient Load Balancing Technique

Hence, an effective load balancing mechanism is the ultimate need of the hour. The unreliability of the centralised load balancer, commonly designed for cloud environments that shows scalability constraints needs to be addressed. To achieve this, an effective distributed architectural design for the serverless edge, aiming to significantly improve the overall system performance needs to be designed [19]. Therefore, addressing load balancing challenges and developing robust solutions remains an active area of research, crucial for ensuring the continued success and smooth operation of cloud-based applications.

Efficient techniques for load balancing are essential for maximising system performance and resource usage in cloud environments. Through the dynamic distribution of workloads among multiple nodes, these algorithms guarantee effective resource allocation and timely service provisioning. Fundamentals of cloud computing, including its design and underlying technologies for virtualization. After that, it discusses load balancing and highlights the difficulties that come with it in a cloud setting. An overview of the many loads balancing algorithms now in use, opening the door to a more thorough comprehension of this crucial cloud computing feature [20]. Traditional load balancing strategies in web server clusters are challenged by big data demands.

To overcome these drawbacks, this research suggests a unique dynamic model that makes use of the SSAWF mechanism [10]. In cloud computing, LBAs present issues and opportunities that are explored in this paper [21]. The advantages and disadvantages of different algorithms are covered, along with frequent load balancing issues and mitigation strategies. They stress that in order to prevent VM overload or underload and to maximise resource utilisation and energy efficiency, LBAs are essential. In order to efficiently manage workloads across several computers, the study emphasises the significance of building LBAs specifically for distributed and parallel cloud systems [4]. The difficulties of load balancing in cloud computing are examined in this paper. The authors acknowledge the various services and advantages that cloud computing provides, but they concentrate on the specifics of load balancing in this context. The paper begins by laying out a basic understanding of cloud computing technology, delves into load balancing strategies, and then highlights some of the major issues in this field [17]. This paper investigates load balancing issues in cloud computing. As cloud services become more popular, effective use of resources and task completion become essential. Although there are many strategies for managing performance, scheduling, and resources, load balancing is still a major obstacle to keeping VMs from overloading and under-loading [22]. The authors stress the necessity for researchers to create LBAs for cloud systems that are better and more effective [18]

2.3|Dynamic Load Balancing and its benefits

Strong suspend functionality is integrated into SSAWF to momentarily halt operations during peak loads. Dynamic load-balancing decisions are made using a cubic exponential smoothing forecast with AHP. In comparison to conventional methods, the authors anticipate that this model will enhance performance under anomalous loads, high concurrency, and high system loads. To fully assess the model's efficacy, more investigation is required. This investigation should focus on a theoretical examination of the SSAWF mechanism, real-world testing, and comparison analysis [10].

Multi-core CPUs become increasingly important as future real-time applications require better performance levels. But to fully utilise these cores, scheduling needs to give equal weight to load balancing (an even distribution of demand) and scheduling (a guaranteed task completion) [23]. In order to achieve both, this paper suggests a unique scheduling algorithm that outperforms current approaches to load balancing while preserving good schedulability. This well-rounded strategy highlights its potential for use in next-generation real-time systems while also reducing energy consumption [9]. *Table 1*. Shows cloud load balancing in real-time applications.

Table 1. Cloud load balancing in real-time applications: a summary.

S/N	Questions	Options
1	The increasing demand for real-time applications and big data processing necessitates efficient load balancing techniques.	[2]
2	Traditional load balancing methods struggle to handle the challenges presented by big data, such as high volume and complex data structures.	[10]
3	Dynamic load balancing adapts to changing workloads by dynamically distributing tasks across available resources, improving performance and system responsiveness.	[10]
4	Various techniques, such as the Self-Scheduling with Adaptive Fuzzy Weights (SSAWF) mechanism and middleware services, are employed for dynamic load balancing in real-time systems.	[10], [4]
5	Cloud computing enables on-demand access to shared resources like software, data, and storage.	[11]
6	Effective load balancing is crucial for managing workloads and ensuring smooth operation in cloud environments.	[11]
7	The development and implementation of efficient LBAs are essential for optimal resource utilisation, performance, and scalability in cloud systems.	[4]

Table 1. Continue.

S/N	Questions	Options
8	Implementing effective dynamic load balancing comes with challenges, including maximizing resource utilization, minimizing data transfer overhead, and efficiently migrating tasks between resources.	[15]
9	Careful consideration needs to be given to task characteristics and network conditions when choosing and implementing dynamic LBAs to ensure efficient task migration.	[15]
10	Ongoing research is focused on addressing challenges and developing robust solutions to ensure the continued success and smooth operation of cloud-based applications.	[17]

A fault-tolerant communication protocol for real-time systems is proposed in this paper. For effective data interchange, it makes use of hierarchical rings and synchronous communication. On top of this protocol, dynamic load balancing is put into place with the goal of distributing workload among processing nodes while taking real-time constraints into account. For distributed real-time systems, this combination method provides optimal performance and fault tolerance [12]. The topic of this study is load balancing for real-time systems operating in dynamic settings with erratic workloads. In such cases, traditional static techniques are ineffective. To address these issues, the authors suggest middleware services with dynamic load balancing. These services were put into use and tested in a Navy system as well as a benchmark suite, proving their efficacy in preserving real-time quality of service even in extremely dynamic environments [13].

The effectiveness of a dynamic load balancing method (DLBS) for hypercube networks in multiprocessor systems is examined in this paper. Equitable workload distribution among processors is the goal of DLBS, which enhances reaction time and resource efficiency. In contrast to static techniques, DLBS functions without any prior knowledge about the state of the system or task behaviour. However, there are obstacles to overcome when creating successful DLBS algorithms, such as maximising system utilisation, reducing data transfer, and carefully choosing which activities to migrate. The performance evaluation of DLBS in a particular network topology is the main objective of this study [15].

3| Proposed Study

3.1| Research Gap

While current research fills in many important gaps regarding cloud load balancing for real-time applications, it still offers essential insights on how the current literature failed to deliver the following: Though generally successful, traditional approaches have trouble preserving fault tolerance in real-time situations, which could cause performance reduction when faults occur [17]. This causes a huge problem in data consistency and hence the overall efficiency of the system falters making it inefficient for any real-time applications. Moreover, maintaining data coherency in the event of a system breakdown continues to be a difficult task in the same context [24].

An additional deficiency is the absence of multi-objective load balancing strategies that take efficiency and security into account. The performance-energy optimization trade-off in the pre-mentioned phase is the main focus of current research, and security is a critical component that is almost always ignored [1]. Given the veracious variety of the huge volume of Big Data being produced at a great velocity, a security threat to it proves to be a major drawback that makes the system inefficient and unworthy as a whole. The literature study currently browsed, also emphasises the absolute and effective necessity of investigating dynamic load balancing strategies further in order to develop solutions that can adjust to the constantly shifting demands of real-time workloads [10]. Prospective directions for future research may also arise from examining the possibilities of artificial intelligence and machine learning for intelligent load balancing in real-time applications [3].

3.2 | Finding

The identification of the research gap underscores the necessity for developing advanced LBAs that are inherently designed to support the unique requirements of existing real-time systems, with a particular emphasis on fault tolerance [4]. Achieving a high degree of system and data integrity, particularly during instances of complete system failures, necessitates the refinement of load balancing methods. Such advancements are critical at this juncture [13].

While traditional approaches like Byzantine Fault Tolerance algorithms and replicated state machines offer some promise, their reliability across diverse scenarios remains questionable, underscoring the need for more stable solutions [1]. The quest for superior resource management, heightened performance, and robust defence mechanisms against cybersecurity threats underlines the importance of integrating security parameters within load balancing frameworks.

Moreover, it is imperative to recognize that while the balancing act between performance enhancement and energy conservation is vital, an overemphasis on this aspect might overshadow other crucial optimization objectives, including security considerations. The development of LBAs that strike a harmonious balance across a spectrum of optimization strategies unveils significant research avenues and presents a viable answer to the challenges outlined.

This approach requires a broadened perspective that extends beyond mere performance and energy efficiency to include considerations such as cost-effectiveness, security, scalability, and judicious resource allocation [2]. By addressing these research voids and experimenting with innovative strategies, we can substantially elevate the effectiveness and robustness of cloud-based load balancing solutions tailored for real-time applications.

This comprehensive examination and proposed strategic redirection aim to catalyse the development of load balancing solutions that are not only responsive to the dynamic demands of real-time applications but also encapsulate a holistic view of system optimization. The incorporation of artificial intelligence and machine learning techniques into load balancing strategies emerges as a particularly promising area of exploration. These technologies have the potential to facilitate the creation of self-adjusting, intelligent systems capable of anticipating and mitigating potential disruptions before they impact system performance [3].

In conclusion, such enhancements are poised to ensure reliable system performance, maintain data consistency, and fortify resilience against a diverse array of challenges. Ultimately, this will contribute to a more effective and efficient resolution of the identified issues, fostering an environment where real-time applications can thrive unhindered by the limitations currently faced.

By embracing this multidimensional approach, future research has the potential to revolutionise the landscape of cloud computing, ensuring that real-time applications are supported by a foundation that is not only efficient and scalable but also resilient and secure against an evolving landscape of threats and challenges. This endeavour will require a concerted effort from the research community to bridge the identified gaps, leveraging the synergies between technological innovation, strategic insight, and a deep understanding of the nuanced requirements of real-time cloud computing environments.

3.3 | Analysis

3.3.1 | Overview of the environment

The foundation of our research on dynamic load balancing for real-time applications in cloud computing is predicated on a meticulously designed experimental environment. This environment is crafted to simulate real-world cloud computing scenarios where load balancing plays a critical role in managing the distribution of computational tasks across a network of virtual servers. The primary aim is to explore the efficiency, fault tolerance, and adaptability of various load balancing strategies under conditions that closely mimic the operational dynamics encountered in live cloud-based applications.

3.3.2 | Hardware specifications

The hardware requirements offer a solid framework for creating an environment that is durable and scalable for load balancing real-time applications. Changes could be made in response to particular needs and financial limitations. In order to handle potential increases in application demand, scalability alternatives must also be taken into account. The following hardware specifications are extremely important for setting up the desired environment:

I. Compute optimised instances: high-performance servers with multiple core CPUs (such as Intel Xeon) or GPUs (e.g., NVIDIA Tesla V100) for parallel processing capabilities essential for training models.

II. Memory: at least 32 GB RAM to handle large datasets and facilitate in-memory data processing.

III. Network: quick networking hardware (such as gigabit switches) to enable seamless data transmission.

IV. Storage: high-speed SSDs for faster read/write operations, with adequate storage capacity to store large datasets, models, and logs.

3.3.3 | Software and tools

The components required to set up, manage, watch over, and secure load balancing infrastructures for real-time applications are provided by these tools and software. The choice of particular tools may differ depending on personal needs, inclinations, and infrastructure compatibility. A suite of software and tools is essential for an efficient machine learning workflow:

I. Data science platforms: JupyterLab or RStudio for interactive data exploration and model development.

II. Machine learning frameworks: tensorflow, PyTorch, or Scikit-learn for building and training machine learning models.

III. Version control: git for code versioning, along with platforms like GitHub or GitLab for repository hosting and collaboration.

IV. CI/CD Tools: jenkins or GitLab CI/CD for automating the testing and deployment of models.

V. Containerization: Docker for creating reproducible environments and Kubernetes for orchestrating containerized applications.

3.3.4 | Configuration details

You can set up and customise HAProxy as a dependable and effective load balancer for your real-time applications, guaranteeing top performance and scalability, by adhering to these setup settings. Changes could be made in accordance with particular needs and best practices in your setting.

I. Development environments: install and configure JupyterLab or RStudio on computer instances or local machines.

II. Machine learning frameworks: ensure the correct installation of TensorFlow, PyTorch, and other libraries, considering the compatibility with CPU or GPU hardware.

III. CI/CD Pipelines: configure pipelines for automated testing and deployment of models, including stages for data validation, model training, evaluation, and deployment.

3.3.5 | Dataset description

Although a specific dataset isn't offered, building, implementing, and optimising the load balancing environment successfully requires having access to pertinent data and performance indicators. The description of this dataset emphasises how crucial it is to comprehend the nature and demands of the workload in order to guarantee the load balancing infrastructure operates as intended. The dataset(s) used in this environment should be:

I. Relevant and diverse: covering a wide range of scenarios and variations to train robust models.

II. Pre-processed and cleaned: including necessary preprocessing steps like normalisation, encoding, and handling missing values.

III. Split appropriately: divided into training, validation, and test sets to ensure proper model evaluation and testing.

3.3.6|Replication instructions

You can build up a load balancing system using HAProxy to effectively spread incoming traffic among numerous backend servers for real-time applications by following these replication instructions. Be careful to modify the configuration and settings in accordance with your unique needs and surroundings. To replicate the environment, follow these steps:

I. Provision hardware: set up compute instances with the specified hardware or utilise cloud-based services.

II. Install software: install and configure the required software and tools on your machines or instances.

III. Prepare data: process and prepare your datasets as described, ensuring they are ready for use in model training.

IV. Develop and train models: utilise the development environment and machine learning frameworks to create and train models.

V. Deploy models: follow the CI/CD pipeline configuration to automate the deployment of models into production.

3.3.7| Troubleshooting and tips

Through adherence to these troubleshooting guidelines and best practices, you may proactively tackle issues and enhance the dependability and efficiency of your load balancing setup for real-time applications. To keep an infrastructure strong and resilient, regular testing, monitoring, and improvement are essential.

I. Resource utilisation: monitor CPU/GPU utilisation and memory usage to optimise the computational resources.

II. Model training efficiency: experiment with different model architectures and parameters to find the optimal balance between performance and training time.

III. Deployment challenges: test the model thoroughly in a staging environment to catch any deployment issues before rolling out to production.

In essence, the setup of our environment is meticulously designed to establish a practical and regulated infrastructure for exploring the challenges and opportunities associated with the implementation of dynamic load balancing within cloud computing. By adopting this detailed approach, our goal is to offer valuable insights that can enhance the efficiency, resilience, and security of real-time cloud applications, thereby paving the way for advancement. This advancement results in quicker iteration, improved model performance, and ultimately, the development of more impactful machine learning applications. It is crucial to continuously learn and adapt to new tools and methodologies in order to uphold an efficient machine learning ecosystem.

For real-time applications to provide high availability, scalability, and optimal performance, an efficient load balancing solution must be implemented. Organisations can avoid load balancing difficulties and ensure a flawless user experience for their real-time apps by adhering to the recommended best practices and environmental configuration. Maintaining the load balancing infrastructure's efficacy over time requires regular monitoring, troubleshooting, and tuning.

Eventually, enterprises may reduce load balancing issues and provide a flawless user experience for their real-time apps by putting into practice the environmental configuration described here. Our applications' success and our users' delight are supported when we have a strong load balancing solution in place that opens the door for improved scalability, reliability, and performance.

As it sets out on its voyage, the EnhancedLoadBalancing algorithm carefully constructs an advanced load balancing environment so that it can distribute incoming network traffic among a fleet of backend servers in an intelligent manner. The foundation for striking a harmonic balance between optimising resource consumption efficiency and preserving high availability and fault tolerance of web services is laid during this crucial preparatory phase. In this first stage, the algorithm takes a number of calculated actions to provide the load balancer with the data and tools it needs to function properly.

The first step is to add a complete list of backend servers to the load balancer. Specific attributes, such weights and capacities, are marked on each server in this list to indicate how well it can handle requests. These characteristics are essential for the load balancer to be able to decide how to divide incoming traffic among the servers according to their individual capacities and load levels.

4 | Algorithm

```
Algorithm enhanced load balancing
Initialize LoadBalancer
LoadBalancer.servers = List of BackendServers with weights and capacities
LoadBalancer.monitoring System = Setup Monitoring System()
PerformInitialHealthChecks (Load Balancer.servers)
Function Handle Incoming Request (Request)
Server = SelectServer (Request, LoadBalancer.servers)
 if CheckServerHealth (Server) then
ForwardRequest(Server, Request)
MonitorPerformance (Load Balancer.servers)
DynamicallyAdjustLoad Balancing Strategy (LoadBalancer.servers)
          else
Retry with next available server or return error
Function SelectServer (Request, Servers)
// Hybrid selection based on Round-Robin, Least Connections, and Weighted
Selected Server = ApplyHybridSelection Algorithm (Servers)
return SelectedServer
Function CheckServerHealth (Server)
// Enhanced with predictive analytics
HealthStatus = PerformHealthCheck (Server)
return HealthStatus == Healthy
Function ForwardRequest(Server, Request)
EstablishSecureConnection (Server)
SendRequestToServer(Server, Request)
Function MonitorPerformance (Servers)
Metrics = CollectPerformanceMetrics (Servers)
```

```
Analyze MetricsForAnomalies (Metrics)
AdjustServerWeights Based On Performance (Metrics)
Function Dynamically AdjustLoad Balancing Strategy (Servers)
ApplyMachineLearning Models For Prediction (Servers)
 AdjustLoadBalancingParameters (Servers)
// Initialization of monitoring and health check systems, not fully detailed
Function Setup Monitoring System()
return new MonitoringSystem
Function PerformInitialHealthChecks (Servers)
foreach Server in Servers do
if not CheckServerHealth (Server) then
MarkServerAsUnhealthy(Server)
Function ApplyHybridSelectionAlgorithm (Servers)
// Logic to combine Round-Robin, Least Connections, and Weighted algorithms
// Placeholder for simplicity
return Server
Function PerformHealthCheck (Server)
// Placeholder for health check logic
return ServerStatus
Function Establish SecureConnection (Server)
// Logic to establish a secure channel
return Connection
Function SendRequestToServer(Server, Request)
// Logic to send the request over the established connection
Function CollectPerformanceMetrics (Servers)
// Collect and return metrics
return Metrics
Function Analyze Metrics For Anomalies (Metrics)
// Analyze metrics using AI/ML for anomaly detection
Function AdjustServerWeights Based On Performance (Metrics)
// Adjust weights dynamically based on performance analysis
Function Apply MachineLearning Models ForPrediction (Servers)
// Use ML models to predict load and adjust strategies
Function AdjustLoad BalancingParameters (Servers)
// Adjust parameters like weights, selection strategy based on predictions
```

End Algorithm.

The algorithm not only creates the list of backend servers but also starts the monitoring system's deployment. This technology is essential to the load balancing environment since it continuously delivers information on the backend servers' performance and health. The monitoring system allows for the proactive identification of problems that could impact response times and service availability, like server overloads or failures, by gathering and evaluating real-time data.

Moreover, because the algorithm understands how important it is to get off to a strong start, it does preliminary health checks on every server. Before a server is added to the load balancing cycle, it is imperative that it is operational and prepared to take traffic. When a server does not pass these initial health checks, it is marked as unhealthy and cannot accept more traffic until the problems are fixed. This preventive action guarantees that the load balancing process starts with a stable and dependable group of servers and helps prevent possible disruptions in service.

The EnhancedLoadBalancing algorithm generates a dynamic and responsive load balancing environment by means of these painstaking initialization procedures. In addition to being set up to divide incoming requests among available resources in an effective manner, this environment is also ready to adjust in real-time to shifting circumstances and demands, guaranteeing the best possible performance and dependability for online services.

4.1 | Discussion

We can create an algorithm to effectively manage incoming traffic and distribute requests among backend servers based on the needs discussed and the load balancing environment's specifications. The algorithm will take into account variables including the state of the server, the load balancing plan, and data from real-time monitoring. A high-level summary of the suggested algorithm is provided below:

Initialization

I. Initialise the load balancer with a list of backend servers and their respective weights.

II. Set up health checks to monitor the status of backend servers periodically.

Receive incoming requests

Listen for incoming requests on the load balancer's frontend interface.

Select backend server

When a request arrives, select a backend server based on the chosen load balancing algorithm:

I. Round-robin: rotate through the list of backend servers in a sequential manner.

II. Least connections: choose the server with the fewest active connections.

III. Weighted load balancing: consider server weights to prioritise higher-capacity servers.

Check server health

Before forwarding the request, perform a health check on the selected backend server:

I. Verify that the server is responsive and available to handle requests.

II. Exclude unhealthy servers from the pool of candidates for load balancing.

Forward request

If the selected backend server passes the health check, forward the incoming request to it.

Monitor performance metrics

Continuously monitor performance metrics, such as response time, throughput, and server load:

I. Collect real-time data on request rate, response time, and server health.

II. Analyse metrics to identify trends, anomalies, and potential bottlenecks.

Dynamic adjustment

Based on the observed performance metrics, dynamically adjust the load balancing strategy and server weights:

I. Scale up or down server weights to distribute traffic more evenly.

II. Implement failover mechanisms to redirect traffic away from underperforming or unavailable servers.

Logging and alerting

I. Log load balancing decisions, server health checks, and performance metrics for monitoring and troubleshooting purposes.

II. Set up alerts to notify administrators of critical events, such as server failures or exceeding predefined thresholds.

Continuous optimization

I. Regularly review and refine the load balancing configuration and algorithm parameters based on changing application requirements and traffic patterns.

II. Conduct performance testing and benchmarking to validate the effectiveness of load balancing strategies and identify areas for improvement.

By implementing this method, organisations can ensure the reliability and scalability of their real-time systems, optimise resource utilisation, and effectively handle incoming traffic.

This strategic approach not only enhances performance but also allows for streamlined operations in dynamic environments. With the utilisation of such techniques, businesses can adapt efficiently to changing demands and fluctuations in workload. Employing this method enables real-time systems to dynamically adjust their load balancing parameters, ensuring a balanced distribution of tasks and improved network responsiveness.

Continuous monitoring and feedback mechanisms play a crucial role in fine-tuning these capabilities, enabling the system to seamlessly navigate varying conditions. Notably, this method promotes a seamless user experience by guaranteeing consistent performance levels.

The inclusion of a detailed flowchart outlining the proposed algorithm provides a visual aid that elucidates the systematic process guiding optimal system performance in real-time environments. *Fig. 3* represents the schematic flowchart diagram of the same.

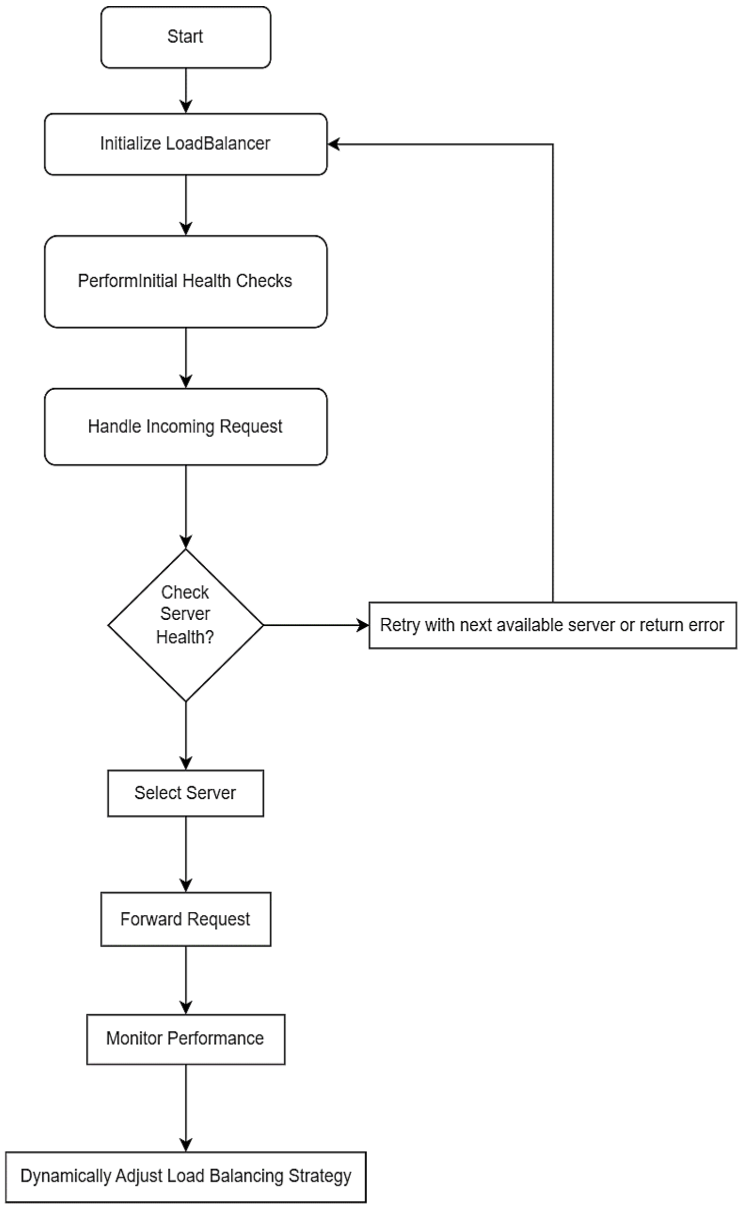
Start
Initialize LoadBalancer
PerformInitial Health Checks
Handle Incoming Request
Check Server Health?
Retry with next available server or return error
Select Server
Forward Request
Monitor Performance
Dynamically Adjust Load Balancing Strategy

Fig. 3. A flowchart depicting the proposed solution.

5 | Conclusion

This paper has therefore delved into the intricacies and demands of load balancing in real-time cloud computing applications, illustrating the problems with generic methods and thus necessitating purpose-built solutions. It establishes that adaptive load balancing mechanisms can significantly improve the efficiency and reliability of real-time systems vis-à-vis the traditional solutions which are one-size-fits-all. This shift will not only see better performance in the system but also enable greater understanding of cloud computer dynamics needed for addressing various needs in digital environments. Therefore, our findings advocate for a change towards customised load balancers, showing that existing approaches fail to address peculiar requirements associated with real-time applications.

The research reveals a major research-practice gap that highlights the importance of formulating load balancing mechanisms that account for dynamic cloud environments and specific requirements of real-time applications. By using adaptive and dynamic approaches we can ensure platforms in cloud computing manage modern complex digital apps hence improving overall system performance and reliability. The study also reveals the future possibility of further research to investigate more dependable and efficient cloud services.

It persuades for joint effort in narrowing down on the answers that can work through the complexities of load balancing in cloud computing especially focusing on real-time applications. The developments made in this field will not only improve Cloud Computing but at the same time play a big role in technological development as it affects both physical and digital landscapes. To sum up, findings from this paper contribute to better understanding of real time cloud computing load balancing issues. Also, it suggests that there should be a paradigm shift towards modularized and adaptable solutions which calls for a holistic approach to address specific needs and changing conditions within a cloud environment. This way, we could meet those challenges head-on and unlock the full potential of cloud computing to ensure its continued role as the driving force behind technology growth while shaping our digital futures.

Author Contributions

Vaidehi Gupta contributed to conceptualization, formal analysis, and drafting the manuscript. Vaibhav Sahay handled methodology, data collection, and contributed to writing – review & editing.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data Availability

Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Kumar, P., & Kumar, R. (2019). Issues and challenges of load balancing techniques in cloud computing: a survey. *ACM computing surveys (CSUR)*, *51*(6), 1–35.

[2] Sun, D., Zhang, C., Gao, S., & Buyya, R. (2024). An adaptive load balancing strategy for stateful join operator in skewed data stream environments. *Future generation computer systems*, *152*, 138–151. https://www.sciencedirect.com/science/article/pii/S0167739X23004016

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

[4] Kang, J., & Waddington, D. G. (2012). Load balancing aware real-time task partitioning in multicore systems. *2012 IEEE international conference on embedded and real-time computing systems and applications* (pp. 404–407). IEEE.

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

[6] El Kabbany, G. F., Wanas, N. M., Hegazi, N. H., & Shaheen, S. I. (2011). A dynamic load balancing framework for real-time applications in message passing systems. *International journal of parallel programming*, *39*, 143–182.

[7] Beltran, M., & Guzman, A. (2008). Designing load balancing algorithms capable of dealing with workload variability. *2008 international symposium on parallel and distributed computing* (pp. 107–114). IEEE.

[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] 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

[10] Aslanpour, M. S., Toosi, A. N., Cheema, M. A., Chhetri, M. B., & Salehi, M. A. (2024). Load balancing for heterogeneous serverless edge computing: a performance-driven and empirical approach. *Future generation computer systems*, *154*, 266–280. https://www.sciencedirect.com/science/article/pii/S0167739X24000207

[11] Mahmood, T., & ur Rehman, U. (2024). Providing decision-making approaches for the assessment and selection of cloud computing using bipolar complex fuzzy Einstein power aggregation operators. *Engineering applications of artificial intelligence*, *129*, 107650. https://www.sciencedirect.com/science/article/pii/S0952197623018341

[12] 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

[13] Liu, D., Shang, W., Zhu, L., & Feng, D. (2016). An improved dynamic load-balancing model. *2016 4th intl conf on applied computing and information technology/3rd intl conf on computational science/intelligence and applied informatics/1st intl conf on big data, cloud computing, data science & engineering (ACIT-CSII-BCD)* (pp. 337–341). IEEE.

[14] Joshi, S., & Kumari, U. (2016). Load balancing in cloud computing: challenges & issues. *2016 2nd international conference on contemporary computing and informatics (IC3I)* (pp. 120–125). IEEE.

[15] 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

[16] Akay, O., & Erciyeş, K. (2003). A dynamic load balancing model for a distributed system. *Mathematical and computational applications*, *8*(3), 353–360.

[17] Sultan, O. H. F., & Khaleel, T. A. (2022). Challenges of load balancing techniques in cloud environment: a review. *Al-rafidain engineering journal (AREJ)*, *27*(2), 227–235.

[18] Welch, L. R., Werme, P. V, Shirazi, B. A., Cavanaugh, C. D., Fontenot, L., Huh, E. N., & Masters, M. W. (2000). Load balancing for dynamic real-time systems. *Cluster computing*, *3*, 125–138.

[19] Shu, B., Chen, H., & Sun, M. (2017). Dynamic load balancing and channel strategy for apache flume collecting real-time data stream. *2017 IEEE international symposium on parallel and distributed processing with applications and 2017 IEEE international conference on ubiquitous computing and communications (ispa/iucc)* (pp. 542–549). DOI: 10.1109/ISPA/IUCC.2017.00089

[20] Bokhari, M. U., Alam, M., & Hasan, F. (2016). Performance analysis of dynamic load balancing algorithm for multiprocessor interconnection network. Perspectives in Science, 8, 564-566.

[21] Domanal, S. G., & Reddy, G. R. M. (2015). Load balancing in cloud environment using a novel hybrid scheduling algorithm. *2015 ieee international conference on cloud computing in emerging markets (CCEM)* (pp. 37–42). IEEE.

[22] Sriram, G. S. (2022). Challenges of cloud compute load balancing algorithms. *International research journal of modernization in engineering technology and science*, *4*(1), 1186–1190.

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

[24] 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