Ensemble Security and Multi-Cloud Load Balancing for Data in Edge-based Computing Applications

Raghunadha Reddi Dornala
Cloud Architect, USA

***Abstract*—Edge computing has gained significant attention in recent years due to its ability to process data closer to the source, resulting in reduced latency and improved performance. However, ensuring data security and efficient data management in edge-based computing applications poses significant challenges. This paper proposes an ensemble security approach and a multi-cloud load-balancing strategy to address these challenges. The ensemble security approach leverages multiple security mechanisms, such as encryption, authentication, and intrusion detection systems, to provide a layered defense against potential threats. By combining these mechanisms, the system can detect and mitigate security breaches at various levels, ensuring the integrity and confidentiality of data in edge-based environments. The multi-cloud load balancing strategy also aims to optimize resource utilization and performance by distributing data processing tasks across multiple cloud service providers. This approach takes advantage of the flexibility and scalability offered by the cloud, allowing for dynamic workload allocation based on factors like network conditions and computational capabilities. To evaluate the effectiveness of the proposed approach, we conducted experiments using a realistic edge-based computing environment. The results demonstrate that the ensemble security approach effectively detects and prevents security threats, while the multi-cloud load balancing strategy with edge computing to improve the overall system performance and resource utilization.**

***Keywords*—Edge computing; cloud computing; dynamic load balancing; fog computing; multi-cloud load balancing**

I. Introduction

With the rapid advancement of edge computing and cloud technology, the aviation industry has witnessed significant transformations in flight management systems. These systems now rely on distributed edge computing infrastructure and leverage multiple cloud service providers for improved performance and reliability [1]. To ensure secure and efficient operations, ensemble security and multi-cloud load balancing techniques have become crucial in managing flight management cloud applications [2]. Ensemble security integrates multiple security measures and protocols to protect flight management systems from cyber threats. As edge computing extends the attack surface, incorporating diverse security mechanisms becomes essential [3]. It can include encryption, authentication, access control, intrusion detection systems, and advanced threat intelligence to safeguard critical flight data and systems [4].

In addition to security, effective load balancing across multiple cloud providers is paramount for flight management cloud applications [5]. Load balancing distributes incoming network traffic across various cloud instances to optimize resource utilization, reduce latency, and enhance system scalability [6]. Multi-cloud load balancing ensures that flight management systems can efficiently utilize cloud resources, handle sudden traffic spikes, and deliver a seamless user experience. Ensemble security and multi-cloud load balancing are intricately linked in edge computing environments [7]. Organizations can dynamically distribute traffic based on security policies and performance metrics by integrating security measures with load-balancing algorithms [8]. It allows flight management cloud applications to balance the workload efficiently across different cloud providers while ensuring data confidentiality, integrity, and availability [9].

This study explores the challenges and opportunities associated with ensemble security and multi-cloud load balancing in the context of flight management cloud applications. We will discuss the essential requirements, potential solutions, and benefits of adopting these techniques in the aviation industry. Furthermore, we will examine real-world use cases and best practices for implementing ensemble security and multi-cloud load balancing to enhance the safety and performance of flight management systems in edge computing environments. By leveraging ensemble security and multi-cloud load balancing with the integration of edge computing with task scheduling, the aviation industry can fully utilize edge computing capabilities while maintaining robust security measures. It ensures flight management systems' safe and efficient operation and opens up new possibilities for advanced applications, such as real-time analytics, predictive maintenance, and intelligent decision-making.

The paper's organization is as follows: Section II presents the literature survey by explaining the existing models and their drawbacks. Section III explained the Intrusion Detection and Prevention System with Multiple Loads Balancing. Section IV introduced edge computing with cloud computing. Section V presented the parameters used to analyze the performance of the proposed approach. Section VI introduced the conclusion for the overall work.

II. Literature Survey

Kuppusamy et al. [10] proposed a combined dynamic model that integrates the reinforced optimization approach to increase the scheduling performance in Fog computing. The experiments are conducted using the FogSim simulator to create the data and an available tool that manages energy efficiency by using the resources in fog computing. The proposed system mainly focused on addressing the schedule

with less cost and analyzing the CPU processing time and assigned memory. Simulation results show that the proposed model performance is increased by 12% to 15% in terms of CPU usage and 6% to 11% of less energy usage for solving job scheduling issues. Jango et al. [11] proposed the two-phase scheduling model consisting of a Bi-factor approach, which classifies the task based on end date and preference scheduling using the advanced Jellyfish Algorithm (ADS). The parameters are measured based on the speed, capacity, task size, complete tasks, and total VM used for resource provisioning fog combined cloud platform. The model was tested on real-time and average datasets considering the small and high workload based on the assigned task. The performance is measured using QoS parameters that reduce the cost and other metrics. K. Cao et al. [12] introduced the edge-based integrated model that deploys the cloud application in heterogeneous servers that estimate the response time from data centers. The proposed model can process both offline and online phases. An edge-based optimized model is executed offline, and the online mobility-based gaming system is developed to overcome the issues. Results show that the response time is improved by 48.57% from the base station. J. Al-Jaroodi et al. [13] proposed a distributed model that combined with cloud and fog-integrated sCPS, named PsCPS. The proposed integrates the multiple clouds, fogs, and sCPS subsystems to provide better services to face many challenges for combining. R. Deng et al. [14] proposed a fog-cloud integrated model that reduces power consumption and measures transmitting delay. The proposed model primarily focused on resolving the workload assignment issue by allocating work between the fog and cloud with minimal usage and artificial service delay. The problem is divided into three sub-issues assigned to sub-systems and solved. The simulation results show that fog computing can significantly improve cloud computing efficiency in terms of bandwidth and dissemination latency. M. Guo et al. [15] introduced the delay-based workload allocation (DBWA) that solves the issues based on energy efficacy and delay-undertake workload allocation issues in IoT-integrated systems. The proposed approach would be improved if the Lyapunov drift-plus-penalty theory was adopted. Finally, the proposed approach obtains better performance in terms of efficient allocation. G. Qu et al. [16] proposed the Deep Meta Reinforcement Learning-based Offloading (DMRO) algorithm that merges various parallel DNNs with Q-learning to make fine-tuned learning approach to solve the adaptive decision-making in the dynamic platform. The proposed approach improved the offloading by up to 18.1%. A. Yousafzai et al. [17] proposed an effective integration-based measurement offloading system. The proposed approach needs the program borders at edge servers that lately combine applications. The outcome saves 45.45% of runtime and 85.45% of energy consumption. The proposed system finally obtained the resource-exhaustive IoT application processing in mobile edge computing (MEC). V. Mohammadian et al. [18] discussed various issues in the cloud load balancing model—the existing approach aimed to find the under-loaded and overloaded nodes and balanced loads. Various issues are identified and solved by the proposed approach in cloud computing. The proposed model obtained better results in terms of fault detection and integrated the simulation tools considered into the account. A. Pandita et al. [19] introduced several scheduling approaches in cloud computing regarding fault tolerance. Several advantages and disadvantages are discussed in the comparison. In cloud computing, the researchers firmly designed an effective fault tolerance system. E. Gures et al. [20] provide the dynamic load balancing approach to solve the generic issue in cloud computing. Various solutions are discussed to solve the load balancing issue. A comparative analysis uses various load-balancing models and analyzes the performance. M. T. Sandikkaya et al. [21] proposed a novel security model (NSM) to secure the PaaS Providers over malicious behavior based on neighbors. The NSM is not like an intrusion detection system, which focuses on processors' usage and challenging resource access. Finally, the NSM approach analyzes the malicious traffic among 100k requests. Several classifiers are used to classify the malicious threads and show better accuracy. O. Sohaib et al. [22] propose a novel 2-tuple fuzzy semantic group selection method that relies on a technology-organization-environment (TOE) system and employs an approach for ordering preference by resemblance to ideal solution (TOPSIS). The TOPSIS is used to help small-to-medium-sized operations assess and make decisions on online e-commerce.

III. Intrusion Detection and Prevention System with Multiple Loads Balancing

An Intrusion Detection and Prevention System (IDPS) mainly focused on detecting and preventing the unauthorized users or malicious activities within a computer network. The following steps involved in IDPS:

1) Data collection: The IDPS collects network traffic and system log data to analyze for potential threats. This can include network packets, log files, and system events.

2) Traffic monitoring: The IDPS monitors network traffic in real-time or analyzes stored data to identify suspicious patterns or anomalies. This can be done using several signature-based and anomaly detection.

3) Signature-based detection (SBD): SBD is mainly compared with the network traffic over system-generated database events, known as attack signatures. If any similarity is identified, then the intrusion is a better one.

$$\text{Match} = \text{Compare}(\text{Signature}, \text{Network Traffic}) \quad (1)$$

4) Anomaly detection: Anomaly detection identifies deviations from normal patterns of network traffic or system behavior. Statistical analysis or machine learning algorithms are often used to detect anomalies.

$$\text{Anomaly} = \text{Compare}(\text{Statistics}, \text{Network Traffic}) \quad (2)$$

5) Behavior-based detection: Behavior-based detection establishes a baseline of normal behavior for the network or system and identifies deviations from that baseline. It can involve monitoring user activity, network connections, or system processes.

$$\text{Deviation} = \text{Compare}(\text{Baseline}, \text{Network}) \quad (3)$$

6) Alert generation: When a potential intrusion is detected, the IDPS generates an alert or notification to notify system administrators or security personnel. The alert may contain information about the detected threat, its severity, and recommended actions.

7) Response and prevention: Based on the severity of the threat, the IDPS may take automated actions to prevent or mitigate the intrusion. This can include blocking suspicious network traffic, terminating malicious processes, or initiating security measures to protect the network.

8) Logging and reporting: The IDPS maintains logs of detected threats, alerts, and responses for future analysis and reporting. These logs can help in understanding attack patterns, identifying vulnerabilities, and improving security measures.

It's important to note that the specific equations or algorithms used in each step may vary depending on the implementation and technology used in the IDPS. The equations mentioned above are generalized representations to illustrate the concept.

IV. Edge Computing with Cloud Computing

Edge computing and cloud computing are two distinct paradigms in computing that serve different purposes and offer unique advantages. However, combined, they can create a robust and efficient computing ecosystem. This section discussed how edge computing and cloud computing work together to enhance the overall computing experience. The cloud platform provides multiple services to users based on the usage of computing resources, data storage, and cloud application over the internet. All these applications and resources are deployed in cloud data centers provided by the cloud service provider (CSP). Users can leverage these resources on demand without needing local infrastructure, reducing costs and improving scalability. Fig. 1 explains the step-by-step process of proposed approach with functionalities.

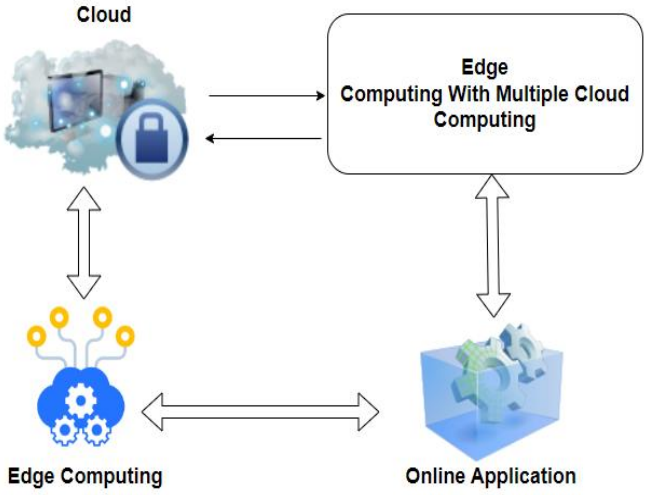

Fig. 1. Proposed system architecture.

On the other hand, edge computing brings computational capabilities closer to the data source or end-users. It involves performing data processing and analysis at the network's edge near the data source rather than sending all data to the cloud for processing. It reduces latency, minimizes bandwidth usage, and enables real-time decision-making, making it ideal for time-sensitive applications and services.

When edge computing and cloud computing are combined, they form a hybrid computing architecture that takes advantage of the strengths of both approaches. For example, sensors, smartphones, and IoT devices can perform initial data processing and filtering at the edge before selectively sending relevant data to the cloud for further analysis or storage. This method reduces the data sent to the cloud, saving bandwidth and allowing faster response times and more efficient resource utilization. The cloud is crucial in this hybrid architecture by providing a centralized and scalable infrastructure for advanced analytics, machine learning, and storage. The data from edge devices can be processed and analyzed in the cloud using sophisticated algorithms and models, taking advantage of the vast computing resources available. The insights and results can be returned to edge devices for immediate action or decision-making.

This edge and cloud computing combination is precious in various industries and applications. For example, edge computing can handle real-time sensor data processing in autonomous vehicles, while the cloud can perform high-level analytics and provide over-the-air updates. In healthcare, edge devices can collect patient data and perform initial analysis, while the cloud can store and process the data for long-term monitoring and research.

A. Load Balancing in Cloud Computing

Load balancing is an essential aspect of cloud computing that helps to optimize resource utilization, improve performance, and ensure the high availability of apps and services. Cloud computing is the delivery of computing resources such as virtual machines, storage, and applications via the internet. Load balancing distributes incoming network traffic across multiple servers or helps to avoid overloading and ensure efficient resource utilization. The primary goal of load balancing in cloud computing is to distribute workload evenly among servers or resources so that no single resource becomes overburdened while others remain underutilized. By evenly distributing the workload, load balancing helps to achieve optimal resource utilization, improve response times, and increase the overall throughput of the system.

Load balancing mechanisms in cloud computing can be categorized into two main types: static and dynamic load balancing. Static load balancing involves distributing the workload evenly based on predefined rules or static configurations. It is suitable for situations where the workload remains relatively constant and predictable. However, in dynamic and unpredictable environments, dynamic load balancing techniques are more effective. Dynamic load balancing utilizes various algorithms and techniques to monitor the system's workload continuously and adjust the distribution of requests accordingly. These techniques take into account factors such as server capacity, response time,

network latency, and current resource utilization to make intelligent decisions about workload distribution. Some popular dynamic load balancing algorithms include Round Robin (RR), Weighted Round Robin (WRR), Least Connections (LC), and Adaptive Load Balancing (ALB).

Load balancers also monitor the health and availability of servers and can automatically reroute traffic away from faulty or overloaded servers, ensuring high availability and fault tolerance. Cloud service providers often offer load balancing services as part of their cloud infrastructure offerings. These load balancers are typically provided as managed services, which means that the service provider takes care of the underlying infrastructure and maintenance, allowing users to focus on their applications and services.

B. Multi-Cloud Load Balancing Model

Round-robin (RR) and Consistent Hashing (CH) are two different algorithms used for load balancing in distributed systems. However, they can also be combined to achieve load balancing with Consistent Hashing.

Round-robin is a simple load balancing algorithm that distributes incoming requests evenly among a set of servers in a cyclic manner. Each request is assigned to the next available server in the rotation. This ensures that each server receives an equal number of requests over time.

For a given request i ($0 \leq i < M$), the server index is calculated as follows:

$$\text{server}_{\text{index}} = i \bmod N \quad (4)$$

Here, "mod" represents the modulo operation. The request is then sent to the server with the corresponding index.

Consistent Hashing is a more advanced load balancing algorithm that provides scalability and minimizes the redistribution of requests when a server is added or removed from the system. It uses a hash function to map each request to a specific server in the system based on the request's key or identifier.

To combine Round-robin with Consistent Hashing for load balancing, we can follow these steps:

Step 1: Initialize a list of servers.

Step 2: Assign each server a unique identifier or key.

Step 3: Create a hash ring to represent the distribution of keys across the servers.

Whenever a request comes in, apply the Consistent Hashing algorithm to determine the server responsible for serving that request.

Step 4: If the server is available and can handle the request, forward it to the server.

If the server is unavailable or overloaded, use Round-robin to select the next available server in the rotation and forward the request to that server.

Step 5: Continue the Round-robin rotation for subsequent requests until a server becomes available again.

Periodically update the hash ring and redistribute the keys when adding or removing servers from the system. By combining Round-robin with Consistent Hashing, we achieve both load balancing and the ability to scale the system dynamically without significant redistribution of requests. Round-robin ensures that the servers receive an equal share of requests, while Consistent Hashing minimizes the impact of adding or removing servers on the overall distribution of requests. This combined approach provides a balanced and efficient load balancing mechanism, ensuring optimal utilization of the server resources in a distributed system.

C. Task Scheduling for Multiple-Cloud

Organizations have increasingly turned to multi-cloud environments in recent years to meet their diverse computing needs. Multi-cloud refers to using multiple cloud service providers simultaneously, allowing businesses to leverage the strengths of different platforms and avoid vendor lock-in. However, managing and optimizing tasks across multiple clouds can be complex and challenging. Task scheduling is a critical aspect of managing cloud resources effectively. It involves determining when and where tasks or workloads should be executed to ensure efficient resource utilization and meet performance objectives. In multi-cloud environments, task scheduling becomes even more intricate as organizations must consider cost, performance, data locality, and compliance across multiple cloud providers. Multi-cloud task scheduling primarily aims to assign tasks to the most appropriate cloud resources based on various criteria, such as workload characteristics, resource availability, and user-defined policies. It involves making intelligent decisions to balance workload distribution, maximize resource utilization, minimize costs, and optimize performance across multiple clouds.

Several key challenges must be addressed to achieve effective task scheduling in a multi-cloud environment. These challenges include:

1) Heterogeneity: Each cloud provider offers different virtual machine types, pricing models, and service-level agreements. Task scheduling algorithms must account for this heterogeneity and make informed decisions about resource selection.

2) Interoperability: Ensuring seamless communication and data transfer between cloud providers is crucial. Task scheduling mechanisms should consider data locality and network latency to minimize data transfer costs and enhance performance.

3) Scalability: Multi-cloud environments often involve a large number of tasks and resources. Task scheduling algorithms must scale efficiently to handle the increased complexity and volume of scheduling decisions.

4) Dynamicity: Cloud environments are dynamic, with fluctuating workloads and resource availability. Task scheduling mechanisms should be adaptable and able to handle workload variations and changes in resource availability in real time.

5) Cost optimization: Multi-cloud environments introduce cost considerations due to varying pricing models and resource utilization. Task scheduling algorithms should minimize costs while meeting performance objectives and user-defined policies.

Addressing these challenges requires the development of intelligent task-scheduling algorithms and frameworks specifically designed for multi-cloud environments. These algorithms should consider workload characteristics, resource availability, performance requirements, and cost constraints to make optimal scheduling decisions.

D. Ensemble Security and Multi-Cloud Load Balancing

Ensemble Security and Multi-Cloud Load Balancing are two important concepts in the field of cloud computing and network security.

E. Ensemble Security

Ensemble Security refers to a comprehensive approach to securing computer networks and systems by using a combination of security measures and tools. The term "ensemble" implies the integration and coordination of various security components to create a stronger and more effective security system.

In ensemble security, multiple security mechanisms are employed to protect against a wide range of threats and vulnerabilities. This paper mainly used the mechanisms which includes intrusion detection and prevention systems (IDPS), data encryption. By combining these different security measures, organizations can create a layered defense system that provides multiple barriers against potential attacks.

The advantage of ensemble security is that it offers defense in depth, meaning that if one security measure fails, others can still provide protection. Additionally, ensemble security enables organizations to leverage the strengths of different security tools and technologies, thereby enhancing overall network security posture.

F. Multi-Cloud Load Balancing (MCLB)

In cloud computing, MCLB is a technique that distributes incoming network traffic across multiple cloud service providers (CSPs) or regions within a single CSP. Load balancing improves application performance, optimizes resource utilization, and ensures cloud-based services' high availability and scalability.

In a multi-cloud environment, organizations may choose to deploy their applications and services across different CSPs to take advantage of each provider's unique capabilities and to mitigate the risk of vendor lock-in. Multi-cloud load balancing allows organizations to distribute the incoming traffic across these different cloud environments, ensuring that the workload is evenly distributed and that no single cloud provider becomes overloaded.

Load balancers serve as go-betweens for users or clients and cloud resources. Incoming requests are intelligently distributed based on factors such as server capacity, latency in the network, and specific to application requirements. This helps in optimizing resource utilization, preventing bottlenecks, and ensuring that the applications remain accessible and responsive. By implementing multi-cloud load balancing, organizations can achieve higher fault tolerance, improved performance, and scalability across multiple cloud environments. It also provides flexibility in managing and scaling resources based on changing demands and enables efficient utilization of cloud resources.

V. Experimental Results

The experiments are conducted on flight management applications which can take Lakhs of requests from the user and managed by the proposed multi-cloud management system with high security. The total requests are analyzed by the proposed model is 10k, 20k and 30k requests. The performance metrics are given below:

1) Response time (RT): It measures the time taken to process a request and provide a response to the client.

$$\mathrm{RT} = \text{Complete time} - \text{request time} \tag{5}$$

2) Throughput (TP): It represents the number of requests processed per unit of time.

$$\mathrm{TP} = \text{Total}\,\frac{\text{Requests}}{\text{Time}} \tag{6}$$

3) Utilization (UTL): It indicates the percentage of resources being used by a server or a load balancer.

$$\mathrm{UTL} = \left(\frac{\text{Resource used}}{\text{Total Resource capacity}}\right) * 100 \tag{7}$$

4) Error rate (ER): It measures the percentage of failed requests or the number of requests resulting in errors.

$$\mathrm{ER} = \left(\frac{\text{Number of failed requests}}{\text{Total number of requests}}\right) * 100 \tag{8}$$

TABLE I. Analysis of Performance Metrics for List of Algorithms for 10K Requests for 10K Files

Algorithms	RT (Sec)	TP (request/sec)	UTL (%)	ER (%)
Least Response Time (LRT)	56.45	67	56	32
IP Hash	51.23	76	61	28
MCLB	45.67	81	67	21

Table I shows the performance of several existing and proposed algorithms based on the given metrics. These metrics mainly shows the huge impact on final outcomes. The proposed model MCLB mainly focused on managing the load balancing among the servers and providing the security for the processing data. For every one request one file is requested by the users. The encryption and decryption is implemented at the data owners end to provide the high end security for the data. The MCLB improved with the integration of Edge computing obtained the performance in terms of given metrics. The total requests processed by the MCLB are 10k requests for 10 files. Fig. 2 shows the performance of algorithm based on the given parameters.

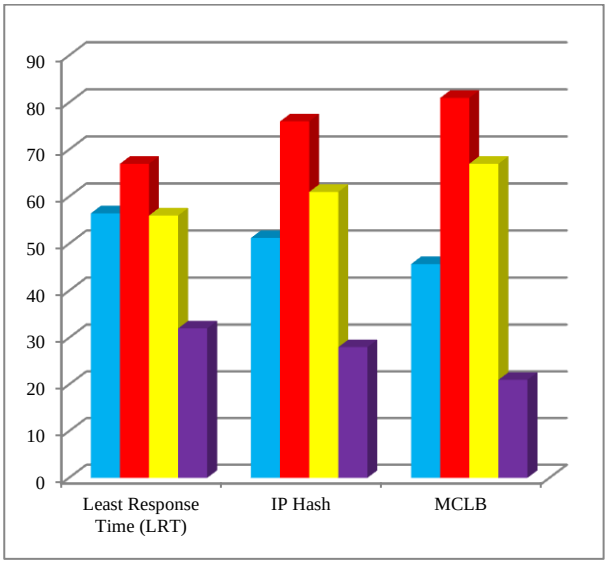

Fig. 2. Performance metrics for list of algorithms for 10k requests.

TABLE II. ANALYSIS OF PERFORMANCE METRICS FOR LIST OF ALGORITHMS FOR 20K REQUESTS

Algorithms	RT (Sec)	TP (request/sec)	UTL (%)	ER (%)
Least Response Time (LRT)	66.45	76	63.4	34.5
IP Hash	61.23	83	65.2	31.2
MCLB	65.67	85.5	69.34	22.2

Table II shows the performance of list of algorithms based on several parameters such as response time (RT) which is high for LRT and low for MCLB. The throughput is low for LRT and high for MCLB which is more efficient. The utilization of resources is high for MCLB and low for LRT. Finally the error rate (ER) shows the low rate for MCLB and high for LRT. Fig. 3 shows the overall performance for 20k requests.

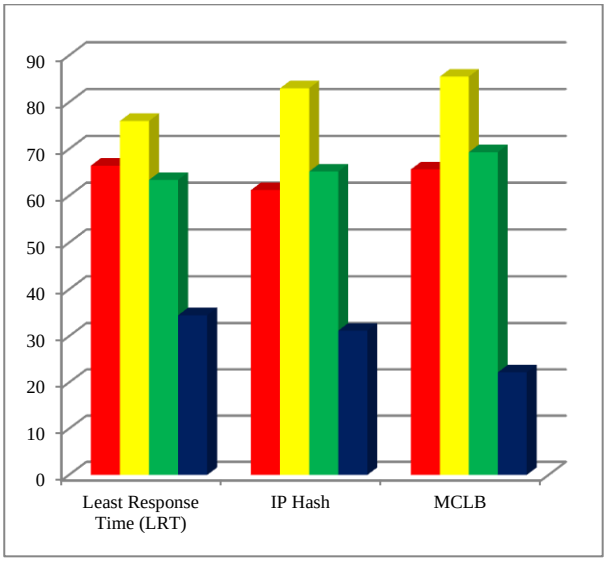

Fig. 3. Performance metrics for list of algorithms for 20k requests.

TABLE III. ANALYSIS OF PERFORMANCE METRICS FOR LIST OF ALGORITHMS FOR 30K REQUESTS

Algorithms	RT (Sec)	TP (request/sec)	UTL (%)	ER (%)
Least Response Time (LRT)	73.55	78.8	64.4	31.5
IP Hash	66.23	84.1	66.2	26.2
MCLB	68.67	86.3	70.34	18.2

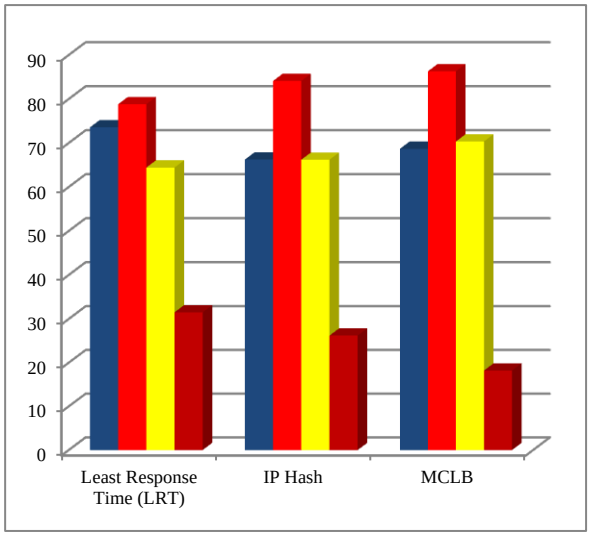

Fig. 4. Performance metrics for list of algorithms for 30k requests.

Table III shows the comparison between LRT, IP Hash and MCLB. The existing model LRT shows the low performance in terms of high RT (Sec), low TP and utilization of resources and high error rate. The proposed approach MCLB shows the better performance in terms of parameters given in Table III. Fig. 4 shows the overall performances for all the algorithms for 30k requests sent by various users present in VMs.

VI. CONCLUSION

Modern cloud computing and network security rely heavily on ensemble security and multi-cloud load balancing. Ensemble security combines multiple security measures to form a robust defense system, whereas multi-cloud load balancing optimizes resource distribution while improving application performance and availability in various cloud environments. The combination of edge computing and cloud computing provides a powerful computing model that takes advantage of the advantages of both paradigms. It allows efficient data processing, lowers latency, increases scalability, and enables real-time decision-making. Organizations can create a robust and flexible computing ecosystem that meets the diverse needs of today's rapidly evolving technological landscape by combining edge computing and cloud computing.

REFERENCES

[1] Zhang, Wei-Zhe & Elgendy, Ibrahim & Hammad, Mohamed & Iliyasu, Abdullah & Du, Xiaojiang & Guizani, Mohsen & Abd El-Latif, Ahmed. (2021). Secure and Optimized Load Balancing for Multi-Tier IoT and Edge-Cloud Computing Systems. IEEE Internet of Things Journal. 1-1. 10.1109/JIOT.2020.3042433.

[2] M. J. Priya and G. Yamuna, "Privacy preserving Data security model for Cloud Computing Technology," 2022 International Conference on Smart Technologies and Systems for Next Generation Computing (ICSTSN), Villupuram, India, 2022, pp. 1-5, doi: 10.1109/ICSTSN53084.2022.9761350.

[3] X. Wei and Y. Wang, "Popularity-Based Data Placement With Load Balancing in Edge Computing," in IEEE Transactions on Cloud Computing, vol. 11, no. 1, pp. 397-411, 1 Jan.-March 2023, doi: 10.1109/TCC.2021.3096467.

[4] A. Kishor, R. Niyogi, A. T. Chronopoulos and A. Y. Zomaya, "Latency and Energy-Aware Load Balancing in Cloud Data Centers: A Bargaining Game Based Approach," in IEEE Transactions on Cloud Computing, vol. 11, no. 1, pp. 927-941, 1 Jan.-March 2023, doi: 10.1109/TCC.2021.3121481.

[5] W. Li, Q. Fan, W. Cui, F. Dang, X. Zhang and C. Dai, "Dynamic Virtual Machine Consolidation Algorithm Based on Balancing Energy Consumption and Quality of Service," in IEEE Access, vol. 10, pp. 80958-80975, 2022, doi: 10.1109/ACCESS.2022.3194514.

[6] Wazir Zada Khan, Ejaz Ahmed, Saqib Hakak, Ibrar Yaqoob, Arif Ahmed, Edge computing: A survey, Future Generation Computer Systems, Volume 97, 2019.

[7] Sulieman, N.A.; Ricciardi Celsi, L.; Li, W.; Zomaya, A.; Villari, M. Edge-Oriented Computing: A Survey on Research and Use Cases. Energies 2022, 15, 452. https://doi.org/10.3390/en15020452

[8] K. Cao, S. Hu, Y. Shi, A. W. Colombo, S. Karnouskos and X. Li, "A Survey on Edge and Edge-Cloud Computing Assisted Cyber-Physical Systems," in IEEE Transactions on Industrial Informatics, vol. 17, no. 11, pp. 7806-7819, Nov. 2021, doi: 10.1109/TII.2021.3073066.

[9] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge computing: Vision and challenges", IEEE Internet Things J., vol. 3, no. 5, pp. 637-646, Oct. 2016.

[10] Kuppusamy, P., Kumari, N.M.J., Alghamdi, W.Y. et al. Job scheduling problem in fog-cloud-based environment using reinforced social spider optimization. J Cloud Comp 11, 99 (2022).

[11] Jangu, N., Raza, Z. Improved Jellyfish Algorithm-based multi-aspect task scheduling model for IoT tasks over fog integrated cloud environment. J Cloud Comp 11, 98 (2022).

[12] K. Cao, L. Li, Y. Cui, T. Wei and S. Hu, "Exploring placement of heterogeneous edge servers for response time minimization in mobile edge-cloud computing", IEEE Trans. Ind. Informat., vol. 17, no. 1, pp. 494-503, Jan. 2021.

[13] J. Al-Jaroodi and N. Mohamed, "PsCPS: A distributed platform for cloud and fog integrated smart cyber-physical systems", IEEE Access, vol. 6, pp. 41432-41449, 2018.

[14] R. Deng, R. Lu, C. Lai, T. Luan and H. Liang, "Optimal workload allocation in fog-cloud computing towards balanced delay and power consumption", IEEE Internet Things J., vol. 3, no. 6, pp. 1171-1181, Dec. 2016.

[15] M. Guo, L. Li and Q. Guan, "Energy-efficient and delay-guaranteed workload allocation in IoT-edge-cloud computing systems", IEEE Access, vol. 7, pp. 78685-78697, 2019.

[16] G. Qu, H. Wu, R. Li, and P. Jiao, "DMRO: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing," IEEE Trans. Netw. Service Manage., vol. 18, no. 3, pp. 3448–3459, Sep. 2021.

[17] A. Yousafzai, I. Yaqoob, M. Imran, A. Gani, and R. M. Noor, "Process migration-based computational offloading framework for IoT-supported mobile edge/cloud computing," IEEE Internet Things J., vol. 7, no. 5, pp. 4171–4182, May 2020, doi: 10.1109/JIOT.2019.2943176.

[18] V. Mohammadian, N. J. Navimipour, M. Hosseinzadeh and A. Darwesh, "Fault-Tolerant Load Balancing in Cloud Computing: A Systematic Literature Review," in IEEE Access, vol. 10, pp. 12714-12731, 2022, doi: 10.1109/ACCESS.2021.3139730.

[19] A. Pandita, P. K. Upadhyay and N. Joshi, "Fault Tolerance Based Comparative Analysis of Scheduling Algorithms in Cloud Computing," 2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET), Kottayam, India, 2018, pp. 1-6, doi: 10.1109/ICCSDET.2018.8821216.

[20] E. Gures, I. Shayea, M. Ergen, M. H. Azmi and A. A. El-Saleh, "Machine Learning-Based Load Balancing Algorithms in Future Heterogeneous Networks: A Survey," in IEEE Access, vol. 10, pp. 37689-37717, 2022, doi: 10.1109/ACCESS.2022.3161511.

[21] M. T. Sandikkaya, Y. Yaslan, and C. D. Ozdemir, "DeMETER in clouds: Detection of malicious external thread execution in runtime with machine learning in PaaS clouds," Cluster Comput., vol. 23, pp. 2565–2578, Dec. 2019.

[22] O. Sohaib, M. Naderpour, W. Hussain, and L. Martinez, "Cloud computing model selection for e-commerce enterprises using a new 2-tuple fuzzy linguistic decision-making method," Comput. Ind. Eng., vol. 132, pp. 47–58, Jun. 2019.