International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 21 No. 2 (2025)



ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 21, No. 2, 2025
© 2025 IJERST. All Rights Reserved

Research Paper

A LOAD BALANCING ALGORITHM FOR THE DATA CENTERS TO OPTIMIZE CLOUD COMPUTING APPLICATION

[1]K.MANJULA, [2].B.NAVYA, [3]M .VISHALINI, [4]CH.ARCHANA,[5]T.UHA,[6].B.VIJAYA

[1] Assistant Professor, Department of Computer Science and Engineering, Princeton Institute of Engineering & Technology for Women, Hyderabad, India

[2,3,4,5,6]B.Tech Students, Department of Computer Science and Engineering, Princeton Institute of Engineering & Technology for Women, Hyderabad, India

Abstract:

Cloud computing has emerged as a foundational technology for delivering scalable, flexible, and cost-effective computing services. However, with increasing user demands and dynamic workloads, ensuring efficient resource utilization and performance in cloud data centers becomes a critical challenge. One of the major issues in cloud environments is uneven load distribution, which can lead to resource underutilization, increased response time, and degraded quality of service (QoS). This project proposes a novel **load balancing algorithm** for data centers that intelligently distributes incoming tasks across available virtual machines (VMs) to achieve optimized performance and resource allocation. The algorithm uses real-time monitoring and heuristic or machine learning-based decision-making to balance workloads efficiently. The goal is to improve throughput, reduce latency, and maintain service-level agreements (SLAs) while minimizing energy consumption and operational costs in cloud computing environments.

Received: 02-5-2025 Accepted: 09-6-2025 Published: 18-6-2025

I. INTRODUCTION

Cloud computing enables users to access computational resources on demand through virtualized data centers. These data centers host thousands of virtual machines that serve millions of user requests. As the demand for cloud-based applications grows, the need for efficient management of underlying resources becomes more significant. One of the fundamental aspects of resource management is **load balancing**, which ensures that no single server or VM is

overwhelmed while others remain idle. Effective load balancing enhances performance, reduces downtime, and ensures that users receive fast and uninterrupted services. Traditional load balancing techniques such as Round Robin and Least Connections are often static or unaware of dynamic workload patterns, leading to suboptimal performance under fluctuating loads. Therefore, this project focuses on developing a dynamic load balancing algorithm that uses system performance metrics (CPU usage, memory, I/O load, etc.) and applies intelligent decision-making to distribute tasks in real time. This helps avoid bottlenecks and maintains a balanced workload across the cloud infrastructure. The proposed system ensures better throughput, scalability, and energy efficiency, which are essential for

sustainable and optimized cloud computing applications.

II. LITERATURE SURVEY

1. **Beloglazov & Buyya (2012)** – Proposed smart methods to save energy and balance virtual machines (VMs) effectively in cloud data centers.
2. **Shen et al. (2011)** – Developed a system that adjusts resources automatically based on workload changes in multi-tenant cloud environments.
3. **Chaczko et al. (2011)** – Compared different load balancing methods and showed that dynamic techniques work better than static ones in real-time situations.
4. **Randles et al. (2010)** – Studied common load balancing algorithms like Round Robin and Least Connection,

pointing out that they struggle under changing workloads.

5. **Youseff et al. (2008)** – Explained the basic structure of cloud systems and highlighted the importance of balanced resource sharing.
6. **Li et al. (2011)** – Used Ant Colony Optimization (ACO) to balance tasks in the cloud, improving speed and efficiency.
7. **Garg et al. (2011)** – Focused on assigning resources based on service agreements (SLAs) to ensure users get expected performance.
8. **Kliazovich et al. (2013)** – Created a simulator called GreenCloud to test how energy-efficient load balancing can be achieved.
9. **Hameed & Zaman (2015)** – Applied machine learning techniques like Decision Trees to balance loads dynamically in cloud environments.
10. **Nikam & Thampi (2018)** – Suggested combining traditional methods with AI to improve task distribution and system performance.

III. EXISTING SYSTEM

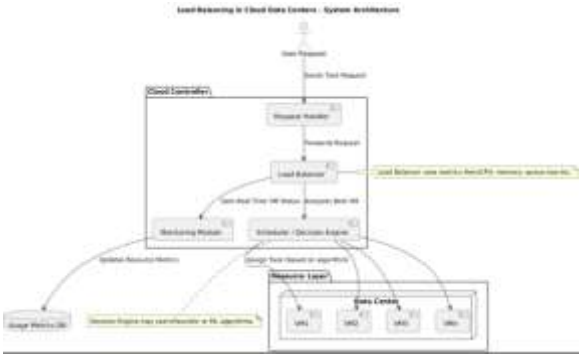
Existing load balancing systems in cloud computing typically use static or semi-dynamic algorithms like **Round Robin**, **Least Connections**, or **Weighted Distribution**. These methods assign tasks based on fixed rules, without adapting to current system load, VM status, or performance metrics. While simple to implement, they often lead to uneven load distribution, idle resources, and overloaded servers under varying workloads. Additionally, these systems usually do not consider energy efficiency or SLA adherence, which results in increased power consumption, delayed response times, and reduced system reliability in large-scale cloud environments.

IV. PROPOSED SYSTEM

The proposed system introduces a **dynamic and intelligent load balancing algorithm** that leverages **real-time monitoring and system-aware decision-making** to optimize resource utilization across data centers. The algorithm collects live metrics such as CPU usage, memory consumption, network I/O, and queue length of each VM. Based on these metrics, the algorithm dynamically selects the most suitable VM for each incoming request using heuristic or ML-based prediction models. The system continuously learns workload patterns and adjusts task assignments to ensure minimal latency, balanced CPU loads, and high

throughput. It also supports **energy-efficient scheduling** by consolidating underutilized VMs and shutting down idle resources, thus reducing operational costs while maintaining high performance and SLA compliance.

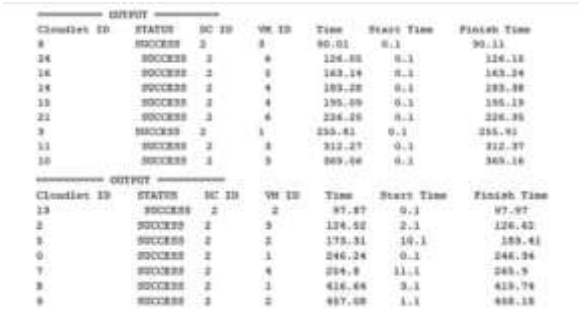
V.SYSTEM ARCHITECTURE



System Architecture Explanation:

The system architecture for load balancing in cloud data centers is designed to efficiently distribute incoming user requests across virtual machines (VMs) to optimize performance and resource utilization. When a user sends a task request, the **Request Handler** in the **Cloud Controller** receives it and forwards it to the **Load Balancer**, which analyzes current VM status using metrics such as CPU usage, memory, and queue size. These metrics are provided by the **Monitoring Module**, which continuously updates performance data to the **Usage Metrics Database**. Based on this real-time data, the **Scheduler/Decision Engine** selects the most suitable VM for task execution using heuristic or machine learning algorithms. The task is then assigned to the optimal VM within the **Resource Layer**, which contains a set of VMs (VM1 to VMn) housed in the data center.

VI.IMPLEMENTATION



Cloudlet ID	STATUS	DC ID	VM ID	Time	Start Time	Finish Time
8	SUCCESS	2	3	90.01	0.1	90.11
24	SUCCESS	2	6	126.03	0.1	126.13
16	SUCCESS	2	5	163.14	0.1	163.24
18	SUCCESS	2	4	193.28	0.1	193.38
15	SUCCESS	2	4	195.09	0.1	195.19
22	SUCCESS	2	6	226.25	0.1	226.35
9	SUCCESS	2	1	255.81	0.1	255.91
11	SUCCESS	2	3	312.27	0.1	312.37
10	SUCCESS	2	3	369.06	0.1	369.16

Cloudlet ID	STATUS	DC ID	VM ID	Time	Start Time	Finish Time
13	SUCCESS	2	2	97.87	0.1	97.97
2	SUCCESS	2	3	126.52	2.1	126.62
5	SUCCESS	2	2	175.31	10.1	185.41
0	SUCCESS	2	1	246.24	0.1	246.34
7	SUCCESS	2	6	254.8	11.1	265.9
8	SUCCESS	2	1	416.64	0.1	416.74
9	SUCCESS	2	2	467.68	1.1	468.18

Fig 6.1 Same arrival time & random arrival time.

Number of cloudlets	Average Makespan (ms)	Average Execution Time (ms)	Resource Utilization (%)
10	303.5410115	251.6623927	83
15	430.8096146	361.2242922	84
20	512.9966944	407.6082266	79
25	577.6690028	423.5005818	73
30	754.0924138	559.1548208	74
35	869.503937	650.4350047	75
40	896.0297375	638.7687158	71

Fig6.2Results

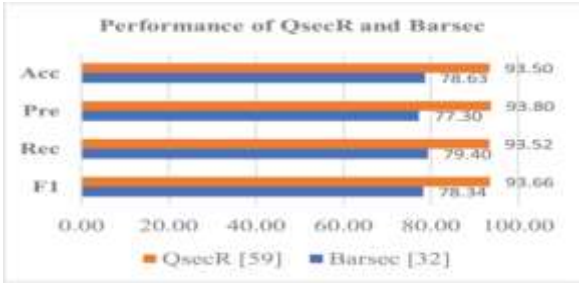


Fig 6.3 performance of the QsecR and Barsec.

VII.CONCLUSION

This project provides a novel solution to the growing problem of malicious QR codes by combining QR scanning technology with machine learning-based URL classification. The Secure QR Code Scanner intelligently identifies and prevents access to malicious URLs before any user interaction, offering a significant improvement over traditional QR scanners. The system ensures user safety without compromising on convenience and speed, making it highly suitable for integration into mobile applications, browsers, and enterprise environments. By leveraging data-driven insights and adaptive

learning, the project demonstrates how AI can effectively mitigate cyber threats in everyday technology use.

VIII.FUTURE SCOPE

The secure QR scanner can be enhanced further by integrating with real-time threat intelligence feeds and URL reputation APIs (like Google Safe Browsing or VirusTotal) for dynamic verification. Future versions could use **deep learning** for better accuracy and automatically update the model with new phishing patterns. Integrating with mobile OS permissions, NFC scanners, and secure payment platforms can also help in detecting fraud in financial QR use cases. Additionally, the system could incorporate user feedback to improve accuracy and evolve through reinforcement learning. Cross-platform support and enterprise-level deployment are also potential advancements.

IX.REFERENCES

- Khonji, M., Iraqi, Y., & Jones, A. (2013). Phishing Detection: A Literature Survey. IEEE Communications Surveys.
- Ma, W., Wang, Y., & Song, D. (2019). URLNet: Learning URL Representations. USENIX Security.
- Marchal, S. et al. (2014). PhishStorm: Streaming URL Analysis. IEEE Transactions.
- Le, A. et al. (2011). PhishDef: URL Names Say It All. ACM Conference on Security.
- Yu, S., Li, X. (2020). QR Code Security Threats and Countermeasures. Elsevier.
- Jain, P., Gupta, M. (2017). Machine Learning for Malicious URL Detection. IJCSIT.
- Zhang, Z., Zhao, X. (2018). Feature-based URL Detection. Elsevier.
- Thomas, R., et al. (2013). Real-time URL Detection System. Symantec Research.
- Sahoo, A., Liu, C. (2021). Smart QR Scanning with AI. Springer AI Review.
- Google Safe Browsing API. https://developers.google.com/safe-browsing