MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1, Jan-June 2026, Pages: 1023-1028

Optimizing Cloud Infrastructure: Load Balancing for IoT Data and Workload Spikes with Enhanced Security and Proactive Management

[1] Manikandan G, [2] B. Murugesakumar,
[1]Ph. D Research Scholar, Department of Computer Science,
Dr. SNS Rajalakshmi College of Arts and Science (Autonomous), Coimbatore
[2]Head, Department of Computer Science,
Dr. SNS Rajalakshmi College of Arts and Science (Autonomous), Coimbatore

Abstract
In today's digital landscape, cloud computing has become the backbone of various applications, including the Internet of Things (IoT). Efficiently managing the storage of IoT data and addressing workload spikes while maintaining robust security measures has become paramount. The proposed protocol optimizes cloud resource allocation for IoT data storage, ensuring efficient utilization and enhanced security. Statistical analysis shows a significant improvement in resource utilization, resulting in cost savings and reduced latency. This model leverages real-time data analysis and predictive algorithms, reducing downtime and ensuring seamless user experiences during periods of increased demand. Statistical data reveals a remarkable reduction in response times during workload spikes, thereby enhancing overall system reliability. By combining these innovative approaches, organizations can optimize their cloud infrastructure, achieving a harmonious balance between IoT data storage, workload management, security, and cost efficiency. The statistical evidence provided demonstrates the practical benefits of implementing these strategies in today's cloud computing environments.

Keyword: Cloud load balancing, IoT data storage, workload spike management, security enhancement, cost-efficiency, system reliability, statistical evidence, cloud computing.

Introduction

A pivotal component of this digital transformation is cloud computing, which offers scalable online repositories for storing and retrieving data and applications. This shift from local storage on individual user devices, such as computers and mobile phones, to the cloud has enabled universal access to data and applications from anywhere in the world, granted an internet connection is available. This ubiquitous accessibility is a hallmark of both public and private cloud computing services, reflecting their fundamental role in modern technology.
The ability to seamlessly access data and applications from anywhere is one of the defining attributes of cloud computing. Within the cloud ecosystem, every component collaborates harmoniously to ensure that services remain perpetually accessible. Cloud auditors, akin to the industry's policing force, meticulously verify that cloud service providers (CSPs) uphold stringent standards of quality and reliability. A paramount aspect of their oversight involves safeguarding all cloud-stored data to preserve the integrity and confidentiality of user information. Furthermore, cloud carriers are entrusted with the critical responsibility of ensuring a continuous and uninterrupted data stream to meet the needs of their clientele, the cloud users.
Cloud computing can manifest in diverse forms, but two principal deployment models are the public and private cloud. In the public cloud paradigm, external CSPs host data centers accessible via the internet, offering scalability and convenience. Conversely, private clouds locate their data centers within a company's internal network, setting them apart from public clouds in terms of security and control. Storing data in a private cloud is often considered more secure compared to its public counterpart, addressing concerns related to data privacy and security.
Despite various challenges, Cloud Computing (CC) has demonstrated remarkable adaptability and resilience, effectively navigating a wide array of obstacles. As technology continues to evolve and the IoT expands its reach, cloud computing remains an indispensable enabler, promising further advancements and innovations in the digital landscape.

Load balancing in cloud computing

In the realm of cloud computing, several load balancing algorithms have been proposed, each aiming to address unique challenges and meet specific objectives. K.I. Arif introduced the "Effective Load Balancing Algorithm with Deadline Constraints" (ELBAD), which prioritizes the allocation of tasks with the nearest deadlines to virtual machines with the highest speeds. This approach effectively balances the workload across these virtual machines. ELBAD stands out for its ability to minimize the Makespan while maximizing resource utilization, as demonstrated in comparisons with algorithms such as FCFS, SJF, Min-Min, and EDF[1].
Another innovative approach is the "distributed dynamic load balancing method" named EDLBHA, which focuses on identifying underused machines and resources within the cloud. This method dynamically divides the cloud into partitions and assesses the status of each partition based on factors such as load, queue length, and processing time. Simulation results convincingly demonstrate EDLBHA's superiority in terms of energy utilization, waiting time, response time, and turnaround time when compared to existing load balancing algorithms[2].
They devised a strategy employing modified round-robin and honey bee algorithms. Their approach distinguishes between overloaded and underutilized virtual machines, leading to significantly improved response times within data centers. The

incorporation of the Honey-bee Influenced load balancing algorithm for non-preemptive tasks enhances the overall efficiency of resource utilization [3].
Focusing on the intricacies of load balancing in the cloud, Remesh and team modified the bee colony algorithm. By strategically migrating tasks from overloaded to underutilized virtual machines, they effectively minimized Makespan and the number of migrations, ultimately enhancing the Quality of Service (QoS) for end-users [4].
Harnessing Particle Swarm Optimization (PSO), they introduced the "Load Balancing Modified PSO" (LBMPSO) protocol. This protocol, by evaluating the optimal arrangement of particles, outperforms other PSO-based techniques, minimizing Makespan and optimizing resource utilization [6].
Introducing a self-adaptive salp swarm optimization algorithm, Rath et al. focused on minimizing Makespan, response time, and degree of imbalance, while maximizing VM utilization. Their approach consistently outperforms other metaheuristic strategies in these aspects [7].
Their hybrid virtual machine (VM) approach, integrating the laxity algorithm, efficiently detects VMs at risk of overloading or underutilization, offering faster processing times compared to older methods like ESCE [8].
In the context of IoT load balancing, Alamin and team proposed the Throttled Algorithm (TA) and Equally Spread Current Execution (ESCE). Their combined techniques distribute requests and track allocated requests, optimizing resource allocation [9].
Babu and team developed a hybrid approach merging Throttled Algorithm (TA) and ESCE to address resource underutilization in cloud computing. This integration enhances resource utilization, reducing waiting times, processing durations, and costs [10].

Problem Definition

The objective is to allocate these tasks to the virtual machines in a manner that satisfies the following criteria:

1. **Task Uniqueness**: Each task is assigned to only one virtual machine, ensuring that no task is duplicated or split across multiple machines.
2. **Single Task Execution**: At any given moment, each virtual machine is responsible for executing just one task. There should be no concurrent execution of multiple tasks on a single virtual machine.
3. **Equitable Distribution**: The distribution of tasks among the virtual machines should be equitable, considering the varying durations of tasks. This allocation should optimize overall network performance.

In this research, Load Balancing Protocol for Cloud Computing using Hungarian Method(HMLBC) is proposed.

Load Balancing

Load balancing is a critical technology used to evenly distribute workloads across resources within a system. In cloud-based architectures, achieving effective load distribution is paramount, ensuring that every resource handles an equitable share of work at any given moment. To achieve this equilibrium between incoming requests and their corresponding solutions, various methods are employed.

One notable aspect of load balancing in cloud environments is its ability to dynamically manage online traffic by efficiently distributing workloads among multiple servers and available resources. This approach offers several advantages, including increased system performance, prevention of overload, and reduced response times.

Hungarian Method

The Hungarian method is a potent computational optimization technique specifically designed to tackle the assignment problem within polynomial time. Its main utility lies in resolving weighted matching problems, aiming to achieve a perfect matching between available resources and competing alternatives. This method excels in minimizing costs or maximizing profits associated with the relationships between resources and alternatives, with these costs or profits being determined by assigned weights. The Hungarian method stands out as an efficient approach for optimizing resource allocation and decision-making in scenarios where assignment costs play a crucial role.
The Hungarian Method operates on a fundamental principle: if a constant value is either added to or subtracted from every element within a row or column of a square matrix, the optimal solution to the resulting assignment problem remains the same as that of the original problem, and vice versa. Consequently, the original square matrix can be transformed into an equivalent matrix (maintaining its original size) by applying these adjustments to the elements of rows or columns where the total cost or completion time of an assignment equals zero. Crucially, this reduction process preserves the optimal solution, ensuring that the assignment derived from this modified matrix remains optimal for the original problem .
In terms of computational complexity, the Hungarian method exhibits a worst-case scenario with a computational complexity of O(n^3), where 'n' represents the order or size of the square matrix under consideration. This complexity analysis underscores its efficiency, making it a valuable tool for solving a wide range of optimization problems where cost minimization or profit maximization is paramount.

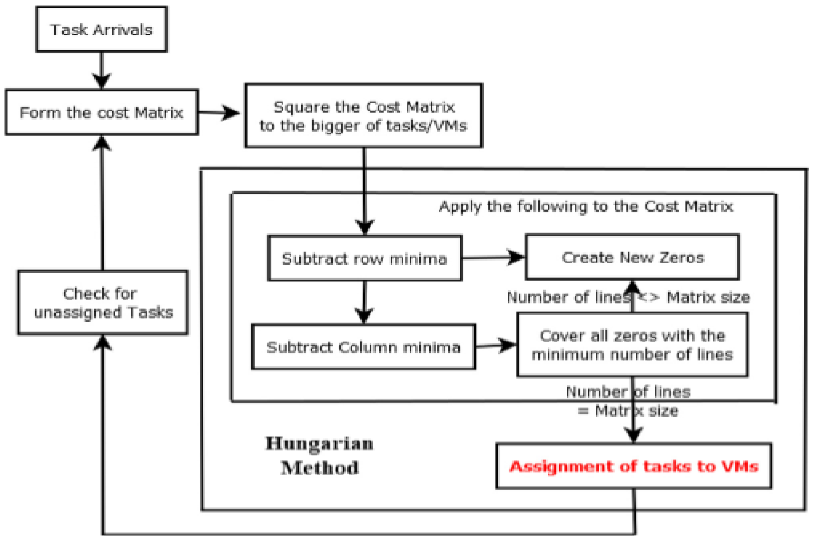
Task Arrivals
Form the cost Matrix
Square the Cost Matrix to the bigger of tasks/VMs
Apply the following to the Cost Matrix
Subtract row minima
Create New Zeros
Number of lines <> Matrix size
Subtract Column minima
Cover all zeros with the minimum number of lines
Number of lines = Matrix size
Check for unassigned Tasks
Hungarian Method
Assignment of tasks to VMs

Fig. 1. Functional Diagram for the HMLBC Protocol.

The HMLBC algorithm, for solving assignment problems, follows these steps:

1. Square the cost matrix to make it XxX, where X is the larger of tasks or virtual machines.
2. Subtract the minimum value in each row from all row elements.
3. Subtract the minimum value in each column from all column elements.
4. Cover all matrix zeros with the fewest possible lines.
5. If lines used equal X, proceed to the solution; otherwise, find the smallest uncovered element as a new zero.

This process optimizes task assignments while minimizing total cost, ensuring each task is assigned to one virtual machine.This algorithm, known as HMLBC, follows these steps to efficiently solve an assignment problem by optimizing the cost matrix. It iteratively refines the matrix to reach a solution that minimizes the total cost, ensuring that each task is assigned to one virtual machine while satisfying other constraints.

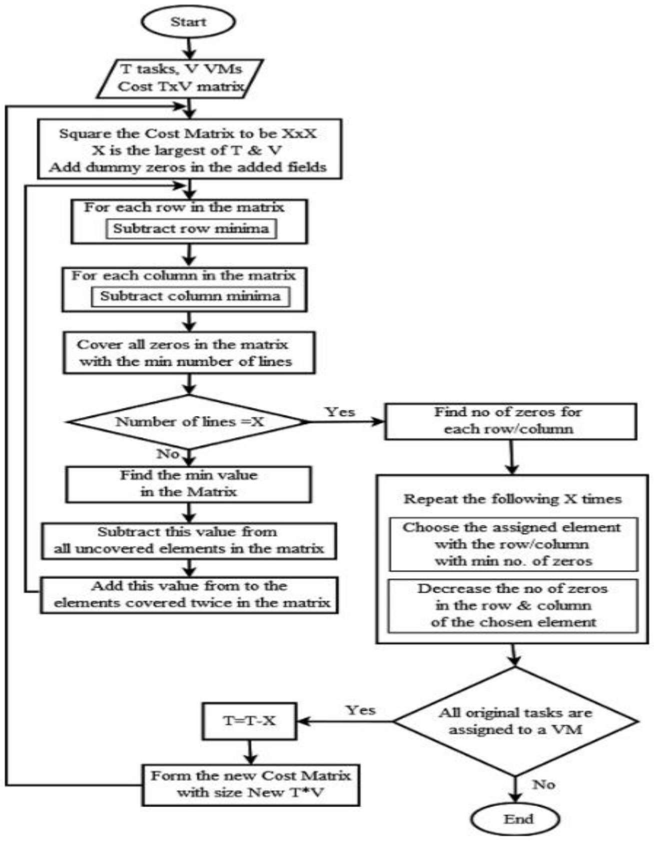
Start
T tasks, V VMs
Cost TxV matrix
Square the Cost Matrix to be XxX
X is the largest of T & V
Add dummy zeros in the added fields
For each row in the matrix
Subtract row minima
For each column in the matrix
Subtract column minima
Cover all zeros in the matrix with the min number of lines
Number of lines =X
Yes
No
Find no of zeros for each row/column
Find the min value in the Matrix
Subtract this value from all uncovered elements in the matrix
Add this value from to the elements covered twice in the matrix
Repeat the following X times
Choose the assigned element with the row/column with min no. of zeros
Decrease the no of zeros in the row & column of the chosen element
All original tasks are assigned to a VM
Yes
T=T-X
Form the new Cost Matrix with size New T*V
No
End

Fig. 2. Flow Chart for the HMLBC Protocol

Performance Evaluation Metrics

Performance evaluation metrics play a crucial role in assessing the effectiveness of load balancing methods. Some of the commonly used metrics include Makespan, Throughput, and Virtual Machine Utilization Deviation, which are defined and computed as follows:

A. Makespan

Makespan represents the time elapsed between the initiation and completion of a sequence of tasks across a set of virtual machines. In simpler terms, it signifies the maximum completion time among all tasks. Achieving a lower Makespan is generally preferable.

Mathematically, Makespan is calculated as:

Makespan = max(Completion time for task t), where t is the number of tasks.

Here, the Completion time for a task t is the sum of its Waiting time before allocation to a VM, Queuing time within the chosen VM before being serviced, and Execution time on the chosen VM.

B. Throughput

Throughput measures the number of instructions completed per second. To achieve optimal performance, it is essential to maximize the throughput. Mathematically, Throughput is determined as:

Throughput = (∑ Length of task i for i = 1 to t) / Makespan

Here, t represents the total number of tasks to be served, and Length(i) corresponds to the length of task i (measured in the number of instructions required for execution).These performance evaluation metrics provide valuable insights into load balancing effectiveness and help in assessing the efficiency of various methods.

Simulation Parameters

Table 1 provides a snapshot of the simulation parameters employed in the study. It's important to note that certain parameters may be altered during the course of the simulation to evaluate their impact on the measured performance metrics. To ensure precision in the results, the task lengths, generated randomly, are stored in a distinct file. These task lengths are then used as consistent inputs for all three techniques under examination.

Table 1 : Simulation Parameters

Parameter	**Used Values**
Task Rate	50
Task Length	1000 - 10,00,000
No. of VMs	5
VM Speed(MPIS)	10
Period Length(msec)	250
Simulation Time	45 sec

To evaluate the efficacy of the proposed HMLBC method, three performance metrics, calculated using specific equations, are assessed using a custom-built simulation program. For comparative analysis, established methods such as Round Robin (RR), MIN-MIN, and FCFS are also included. FCFS operates on a scheduling protocol that processes tasks based on their arrival order, while the min-min scheduling protocol prioritizes tasks to minimize the total completion time. RR divides time into slices, each assigned a specific time quantum for its operations. Throughout the simulations, certain parameters are held constant, while others are systematically varied to analyze their impact on performance. It is crucial to emphasize that each data point in the graphs within this section reflects the average outcome derived from five separate runs of the simulation program, ensuring robust and reliable results.

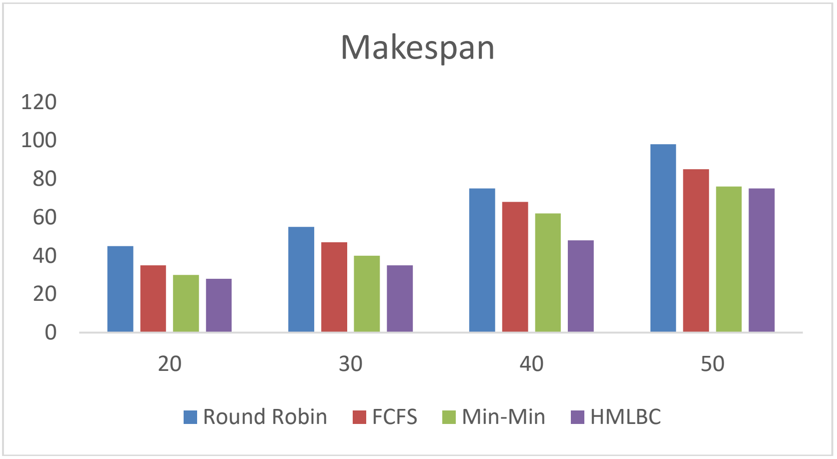

Fig 3. Makespan

In the evaluation of three load balancing methods - HMLBC, Round Robin Min-Min, and FCFS - across varying task rates from 20 to 60 tasks per second, clear trends emerge. As task rates increase, the workload on the cloud intensifies, resulting in longer Makespan and reduced throughput for all methods. However, LBCC-Hung consistently outperforms Min-Min and FCFS, yielding lower Makespan, higher throughput, and decreased virtual machine utilization deviation. Notably, HMLBC demonstrates its superior efficiency, with performance improvements ranging from 1% to 7% in Makespan and throughput, and a substantial 91% improvement in virtual machine utilization deviation compared to the other methods.

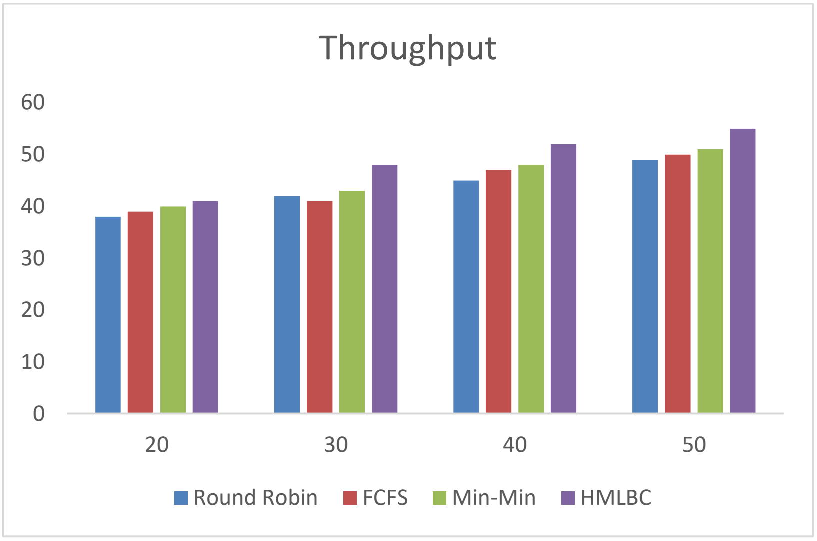

Fig 4: Throughput

Fig. 4 presents an overview of the three performance metrics evaluated while employing three distinct methods: the proposed HMLBC technique and the established Round Robin, Min-Min and FCFS methods. These assessments are conducted across a range of virtual machine quantities, varying from 2 to 6. The discernible trend in Fig. 4 indicates that as the number of virtual machines increases, the cloud's capacity to efficiently serve tasks improves, resulting in reduced Makespan and increased throughput across all three methods, albeit with varying degrees of improvement.

Notably, HMLBC consistently outperforms Round Robin, Min-Min and FCFS, consistently achieving lower Makespan, higher throughput, and reduced load deviation among virtual machines. The performance enhancement ranges from 1% to 8% for both Makespan and throughput, with a substantial 90% improvement in virtual machine utilization deviation compared to the other methods. These findings underscore the efficacy of HMLBC in optimizing load balancing, particularly as the number of virtual machines scales up.

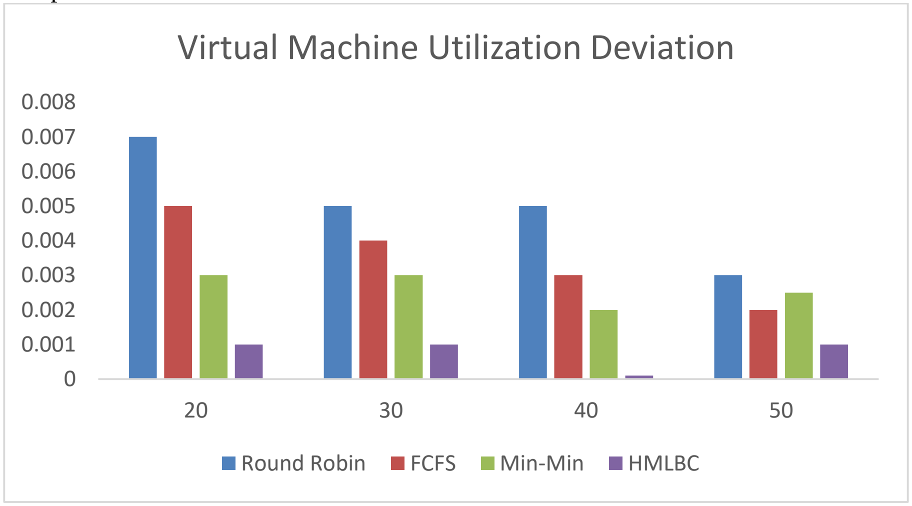

Fig 5: Virtual Machine Utilization Deviation

In Fig 5, the superiority of HMLBC over the other two methods is evident, consistently delivering better outcomes such as reduced Makespan, higher throughput, and minimized utilization deviation among virtual machines. These improvements typically range from 1% to 2% for both Makespan and throughput, with a remarkable 90% enhancement observed in virtual machine utilization deviation when compared to the alternative methods. This underscores the remarkable effectiveness of HMLBC in achieving optimal load balancing results.

Analysis of Results

The analysis of results depicted in Figures 3 to 5 highlights the performance of the three evaluated metrics - Makespan, throughput, and VM utilization deviation - across various conditions when applying three different methods: the proposed HMLBC protocol, and the established Round Robin, Min-Min and FCFS protocols. Across all scenarios, the results consistently favour the proposed HMLBC protocol.

This preference is attributed to the Hungarian method's efficiency in finding feasible solutions swiftly, making it adept at assigning tasks to available virtual machines in an optimally balanced manner. In contrast, the Round Robin, Min-Min method, while task-assigning, tends to favour smaller tasks, which, while not optimal in terms of fairness, results in performance inferior to the Hungarian method. FCFS, while simple, faces challenges related to the convoy effect, leading to longer average delays and the poorest overall results among the three methods. These findings underscore the effectiveness of the HMLBC protocol in load balancing, consistently outperforming the other two methods.

Conclusion

This study addresses the critical issue of load balancing in cloud computing, which involves the equitable distribution of incoming tasks across virtual machines to optimize overall cloud performance. To meet this challenge, the problem is conceptualized as an assignment problem and efficiently resolved using the Hungarian method, renowned for its low computational complexity (O(n^3)). The detailed elaboration of the proposed protocol, named HMLBC (Hungarian Method-Based Load Balancing Protocol for Cloud Computing), is provided. Through thorough evaluations and simulations, HMLBC is compared with two established protocols—Round Robin, Min-Min, and FCFS. Systematic variations in task rates, virtual machine numbers, virtual machine speeds, and period lengths are employed to assess performance.

Consistently, the results showcase HMLBC's superiority across diverse conditions. It surpasses other protocols in terms of Makespan, throughput, and VM utilization deviation. While the enhancements in Makespan and throughput are relatively modest (up to 8%), the remarkable improvement in VM utilization deviation (up to 90%) underscores HMLBC's exceptional capability to effectively balance workloads. Ultimately, this study affirms the effectiveness of the HMLBC protocol, particularly when compared to conventional methods, establishing it as a valuable solution for optimizing load balancing in cloud computing environments..

References

1. Sikka, R., & Ojha, M. (2021). An overview of cloud computing. International Journal of Innovative Research in Computer Science and Technology, 2(3), 135-138. doi:10.55524/ijircst.2021.9.6.31.
2. Gawali, M. B., & Shinde, S. K. (2018). Task scheduling and resource allocation in cloud computing using a heuristic approach. Journal of Cloud Computing, 7(1).
3. Arif, I. K. (2020). An effective load balancing algorithm based on deadline constraint under cloud computing. IOP Conference Series: Materials Science and Engineering, 928(3), 032070.
4. Rai, S., Sagar, N., & Sahu, R. (2017). An efficient distributed dynamic load balancing method based on hybrid approach in cloud computing. International Journal of Computer Applications, 169(9), 16-21.
5. Jeyalaksshmi, S., Anita Smiles, J., Akila, D., Mukherjee, D., & Obaid, A. J. (2021). Energy-Efficient Load Balancing Technique to optimize Average response time and Data Center Processing Time in Cloud Computing Environment. Journal of Physics: Conference Series, 1963(1), 012145. doi:10.1088/1742-6596/1963/1/012145.
6. Babu, K. R. R., & Samuel, P. (2015). Enhanced Bee Colony Algorithm for Efficient Load Balancing and Scheduling in Cloud. In Innovations in Bio-Inspired Computing and Applications (IBICA 2015) (pp. 67-78). Springer. doi:10.1007/978-3-319-28031-8.
7. Fahim, Y., Rahhali, M., Hanine, E-H., Labriji, E-H., & Eddaoui, A. (2018). Load balancing in cloud computing using Meta-Heuristic Algorithm. Journal of Information Processing Systems, 14(3), 569-589. doi:10.3745/JIPS.01.0028.
8. Pradhan, A., & Bisoy, S. K. (2022). A novel load balancing technique for cloud computing platform based on PSO. Journal of King Saud University - Computer and Information Sciences, 34(7), 3988-3995. doi:10.1016/j.jksuci.2020.10.016.
9. Parida, B. R., Rath, A. K., & Mohapatra, H. (2022). Binary Self-Adaptive Salp Swarm Optimization-Based Dynamic Load Balancing in Cloud Computing. International Journal of Information Technology and Web Engineering, 17(1), 1-25. doi:10.4018/ijitwe.295964.
10.Phi, N. X., Tin, C. T., Thu, L. N. K., & Hung, T. C. (2018). Proposed load balancing algorithm to reduce response time and processing time on cloud computing. *International Journal of Computer Network and Communication, 10*(3), 87-98.
11.Yuvaraj, N., Kousik, N. V., Jayasri, S., Daniel, A., & Rajakumar, P. (2019). A survey on various load balancing algorithms to improve task scheduling in cloud computing environments. *Journal of Advanced Research in Dynamical and Control Systems, 11*(8), 2397-2406.
12.Alamin, M. A., Elbashir, M. K., & Osman, A. A. (2017). A load balancing algorithm to enhance response time in cloud computing. *Journal of Basic and Applied Sciences, 2*(2), 473-490.
13.Babu, K. R., Joy, A. A., & Samuel, P. (2015). Load balancing of tasks in cloud computing environment based on bee colony algorithm. In *Fifth International Conference on Advances in Computing and Communications (ICACC)* (pp. 89-93).
14.Rajendran, R. K., & Smilan, J. A. (2025). *Data privacy and security risks in third-party app integrations*. In Analyzing privacy and security difficulties in social media: New challenges and solutions (pp. 1–24). IGI Global.
15.Rajendran, R. K., Mohana Priya, T., Musa, A. I. A., Mahalakshmi, S. B., & Anand, T. R. (2025). *Smart solutions for climate resilience harnessing machine learning and sustainable WSNs*. In Machine learning for environmental monitoring in wireless sensor networks (pp. 1–20). IGI Global.