Cureus Journal of Computer Science
https://doi.org/10.7759/s44389-026-00370-x

Original Article

A Hybrid Deep Learning and Metaheuristic Model for Dynamic Load Balancing in Cloud Data Centers

Michael A. Ibiyomi [1,]

1. *Department of Computer Science, Federal Polytechnic, Ile Oluji, Ile Oluji, NGA*

Received: August 03, 2026 | Review began: August 17, 2026 | Review ended: September 30, 2026 | Published: November 16, 2026

© **Copyright** 2026
This is an open access article distributed under the terms of the Creative Commons Attribution License CC-BY 4.0., which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Abstract

Cloud computing environments face significant challenges in maintaining optimal performance due to dynamic workloads and heterogeneous resource configurations. Traditional load balancing algorithms often fail to adapt to real-time fluctuations, leading to suboptimal resource utilization, increased latency, and degraded performance. This paper presents a novel hybrid load balancing model that integrates Long Short-Term Memory (LSTM) networks with Ant Colony Optimization (ACO) and Throttle algorithms. The proposed approach leverages LSTM's predictive capabilities to forecast workload patterns, ACO's metaheuristic optimization for intelligent task scheduling, and Throttle mechanisms for dynamic resource allocation. Experimental evaluation using CloudSim-based simulation demonstrates that the hybrid model achieves a response time of 120 ms, throughput of 150 tasks/second, and resource utilization of 85%, significantly outperforming configurations lacking individual components. The LSTM model achieved a Mean Absolute Percentage Error of 2.17% and inference time of 0.7249 seconds, confirming its suitability for real-time applications. Ablation studies validate the synergistic contribution of each component, with the complete hybrid model consistently outperforming ablated versions across all performance metrics.

Categories: Cloud Management and Monitoring, Cloud Networking, Machine Learning (ML)

Keywords: cloud computing, load balancing, workload prediction, long short-term memory (lstm), ant colony optimization (aco), cloudsim simulation, resource utilization

Introduction

Cloud computing has fundamentally transformed the delivery and consumption of computing resources by providing on-demand access to shared pools of configurable resources [1], with organizations increasingly transitioning to cloud-based solutions to achieve scalability, cost reduction, and operational flexibility. However, ensuring optimal performance and reliability in cloud environments remains a critical challenge, particularly as workloads become increasingly dynamic and heterogeneous [2].

Load balancing serves as a fundamental mechanism for distributing incoming network traffic and workloads across multiple servers, preventing any single resource from becoming overloaded [3]. Traditional load balancing techniques, including Round Robin, Least Connections, and Weighted Load Balancing, rely on predefined criteria and static assumptions about system state. These methods often struggle to adapt to the rapid and unpredictable workload fluctuations characteristic of contemporary cloud environments [4].

Recent advances in artificial intelligence have opened new possibilities for intelligent load balancing, with deep learning models - particularly Long Short-Term Memory (LSTM) networks - excelling at capturing temporal dependencies in sequential data and forecasting future workload patterns [5]. Metaheuristic algorithms such as Ant Colony Optimization

(ACO) have demonstrated effectiveness in solving complex optimization problems through collective exploration and near-optimal solution discovery [6], while Throttle algorithms complement these approaches by providing real-time adjustments based on system performance metrics.

The Round Robin algorithm distributes requests sequentially across servers without considering individual server capacities or current loads [7], and while simple to implement, this approach can lead to imbalances when servers have heterogeneous capabilities. The Least Connections method directs traffic to servers with the fewest active connections, yet it fails to account for the varying resource demands of different tasks [8].

Kazeem et al. [9] proposed a hybrid MMRR algorithm combining maximum-minimum and round-robin approaches, achieving 89% improvement in response time and cost-effectiveness, though such hybrid approaches still lack predictive capabilities and real-time adaptability. Alsadie [10] introduced TSMGWO, a multi-objective Grey Wolf Optimizer that reduced makespan by up to 67.52% compared to FCFS, while Kishor and Chakarbarty [11] developed a Smart Ant Colony Optimization algorithm for fog computing environments, achieving latency improvements of 12.88% over Round Robin and 6.98% over throttled schedulers. Malti et al. [12] proposed a hybrid optimization combining flower pollination with grey wolf optimizer, demonstrating enhanced makespan, resource utilization, and throughput. While these studies demonstrate the potential of metaheuristics, they typically lack predictive components that anticipate future workload conditions.

Deep learning techniques have increasingly been applied to cloud resource management, with Simaiya et al. [13] proposing a DPSO-GA hybrid model combining deep learning with particle swarm optimization and genetic algorithms for workload prediction. Dastagiraiah et al. [14] introduced DVMPDL, a deep learning framework for dynamic virtual machine provisioning that optimized QoS metrics including throughput, latency, and response times. Manickavasagan et al. [15] demonstrated the Harris Hawk Optimization Algorithm for cloud resource allocation, achieving improved reliability and reduced response times. However, these approaches often lack the dynamic control mechanisms necessary for real-time load balancing adjustments [16].

This paper proposes a hybrid load balancing model that synergistically integrates LSTM networks, ACO, and Throttle algorithms. The key contributions of this study are as follows: (1) Design and implementation of an LSTM-based workload prediction model that achieves 2.17% Mean Absolute Percentage Error (MAPE) for forecasting server loads. (2) Integration of ACO algorithm leveraging LSTM predictions for optimized task scheduling. (3) Incorporation of Throttle mechanism for dynamic resource allocation and system stabilization. (4) Comprehensive evaluation demonstrating significant improvements in response time (120 ms), throughput (150 tasks/second), and resource utilization (85%).

How to cite this article:
Ibiyomi M A (November 16, 2026) A Hybrid Deep Learning and Metaheuristic Model for Dynamic Load Balancing in Cloud Data Centers. Cureus J Comput Sci 3 : es44389-026-00370-x. DOI https://doi.org/10.7759/s44389-026-00370-x

Materials And Methods

System architecture

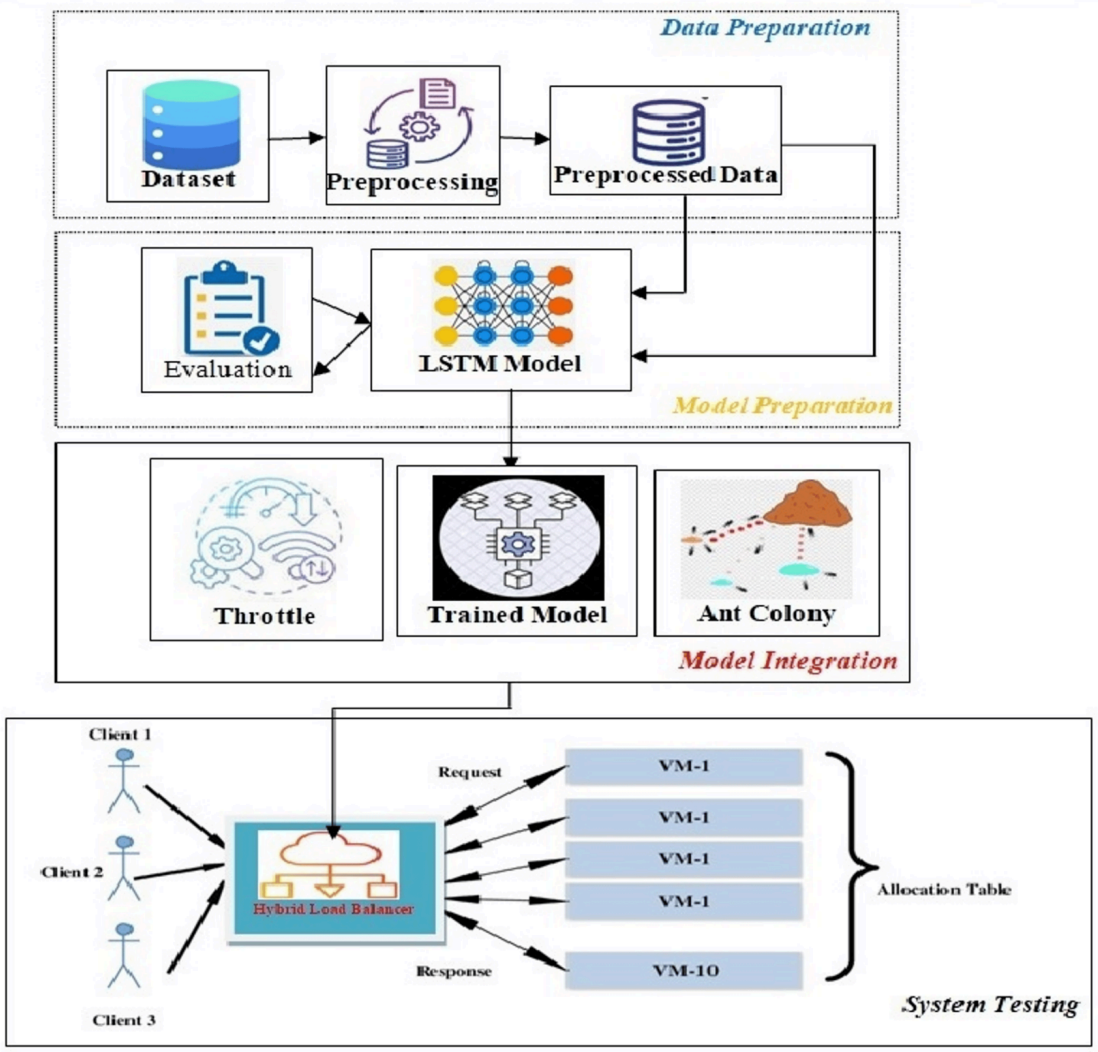


FIGURE 1: System Architecture of Hybrid Load Balancing Model

Source: Adapted from Ibiyomi et al. [16] and developed by the authors. LSTM, Long Short-Term Memory; ACO, Ant Colony Optimization

The proposed hybrid load balancing system extracted from the study [16] comprises four major components: data preparation, model preparation (LSTM, ACO, Throttle), model integration, and system testing. Figure *1* illustrates the overall architecture. The data preparation component collects and preprocesses workload data including CPU utilization, memory usage, disk I/O, and network bandwidth metrics. The model preparation component designs and trains the LSTM predictor, configures the ACO optimizer, and implements Throttle control mechanisms. Model integration combines these components into a cohesive system, while system testing evaluates performance in simulated cloud environments.

Dataset description and preprocessing

The experimental dataset comprises synthetic workload traces generated to simulate cloud data center operations. The dataset [17] includes 10,000 data points capturing five key performance metrics (Table *1*).

Feature	Range	Description
CPU Utilization	0-100%	Percentage of CPU capacity used
Memory Usage	0-16 GB	RAM utilization per server
Disk I/O	0-1000 IOPS	Input/Output operations per second
Network Bandwidth	0-1000 Mbps	Network throughput
Task Arrival Rate	0-50 tasks/second	Incoming task rate

TABLE 1: Dataset Features and Descriptions

CPU, Central Processing Unit

Data preprocessing steps

Normalization

All features were scaled to (0,1) range using Min-Max scaling:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Sequence Creation

Time-series sequences of length 10 were created for LSTM training using sliding window approach

Data Splitting

80% (8,000 samples) for training, 20% (2,000 samples) for testing

LSTM workload prediction model

The LSTM architecture captures temporal dependencies in server load data:

Network Architecture

Input layer: Sequence length of 10 time steps

LSTM Layer 1: 64 units with ReLU activation, return_sequences = True

LSTM Layer 2: 128 units with ReLU activation

Dense Output Layer: 1 unit (predicted load)

LSTM workload prediction

The LSTM model is designed to capture temporal dependencies in server load data. The architecture consists of two LSTM layers (64 and 128 units respectively) followed by a dense output layer.

The LSTM cell operations are mathematically defined as:

$$Inputgate: i_g = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \quad (1)$$

$$Forgetgate: f_g = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \quad (2)$$

$$Outputgate: o_g = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \quad (3)$$

$$Cellstateupdate: c_t = f_t * c_{t-1} + i_g * \tilde{c}_t \quad (4)$$

$$Hiddenstate: h_t = o_g * \tanh(c_t) \quad (5)$$

Where:

W_i, W_f, W_o = weight matrices for input, forget, and output gates

b_i, b_f, b_o = bias vectors

σ = sigmoid activation function $(\sigma(x) = 1/(1 + e^{-x}))$

$tanh$ = hyperbolic tangent activation function

x_t = input at time step t

h_{t-1} = previous hidden state

c_{t-1} = = previous cell state

$\tilde{c}_t$ = candidate cell state $(\tanh(W_c \cdot [h_{t-1}, x_t] + b_c))$

Training Configuration

Optimizer: Adam with learning rate 0.001

Loss function: Mean Squared Error (MSE)

Batch size: 32

Epochs: 20 (with early stopping patience of 3)

Validation split: 20% of training data

Ant Colony Optimization for Task Scheduling

The ACO algorithm leverages LSTM-predicted loads as heuristic information for task scheduling. The probability that ant *k* assigns task *i* to server *j* is:

$$P_{ij}^k = \frac{[T_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [T_{il}]^\alpha \cdot [\eta_{il}]^\beta} \quad (6)$$

The heuristic information incorporates predicted server load:

$$\eta_{ij} = \frac{1}{\hat{L}_j(t)} \quad (7)$$

Parameters:

τ_{ij} = pheromone intensity on path from task i to server j

α = pheromone influence factor (set to 1.0)

β = heuristic information influence factor (set to 2.0)

N_i^k = set of feasible servers for ant k to assign task i

Pheromone updates follow:

$$T_{ij} = (1 - \rho) \cdot T_{ij} + \Delta T_{ij} \quad (8)$$

Where:

ρ = evaporation rate (set to 0.1)

$\Delta\tau_{ij}$ = pheromone deposited by ants

The algorithm initializes ants (15-50 per iteration based on test scenario) and runs for 50-100 iterations to converge to optimal task assignments.

Throttle Algorithm for Dynamic Control

The Throttle algorithm dynamically adjusts resource allocation based on predicted loads and real-time conditions. The throttle factor for server *j* is:

Throttle Factor Calculation:

$$\theta_j = \frac{R_{desired}}{R_j(t)} \tag{9}$$

Desired Resource Calculation:

$$R_{desired} = \frac{\sum_{j=1}^{N} \widehat{R}_j(t)}{N} \tag{10}$$

and

Predicted Resource Need:

$$\widehat{R}_j(t) = R_j(t) + \widehat{L}_j(t) \tag{11}$$

Where:

θ_j = throttle factor for server j

$R_j(t)$ = current resource allocation of server j

$\widehat{L}_j(t)$ = LSTM-predicted load for server j

N = total number of servers

When $\theta_j > 1$: Server requires additional resources
When $\theta_j < 1$: Server has surplus resources

Integrated Framework

The complete hybrid algorithm operates through three sequential phases:

Phase 1 - Prediction:
LSTM model forecasts future server loads $\widehat{L}_j(t+1)$ for the next time interval.

Phase 2 - Optimization:
ACO algorithm determines optimal task-to-server assignments using predicted loads:

Each ant constructs a complete assignment solution.

Solutions are evaluated using objective function: minimize makespan and maximize load balance.

Pheromone trails are updated based on solution quality.

Phase 3 - Control:
Throttle algorithm adjusts resource allocations:

Calculates throttle factors for each server.

Reallocates resources from underutilized to overloaded servers.

Prevents resource over-provisioning and bottlenecks.

This integrated approach enables proactive rather than reactive load balancing, allowing the system to anticipate workload changes and pre-allocate resources accordingly.

Results And Discussion

Results

Simulation Environment

The hybrid model was implemented using Python 3.9 and evaluated using CloudSim 5.0, a widely-used cloud computing simulation framework. All experiments were conducted on a system with the following specifications:

i. Hardware: Intel Core i7-10750H processor, 16GB RAM, NVIDIA GeForce GTX 1650 GPU.

ii. Software: Windows 11 Professional, Python 3.9.7, TensorFlow 2.8.0, NumPy 1.21.0, Pandas 1.3.0.

iii. CloudSim Configuration: Data center with 10-25 virtual machines, 4-100 cloudlets per test scenario.

Dataset Details

The dataset details are as follows: A synthetic dataset comprising 100,000 data points was generated using CloudSim's workload generator, simulating realistic cloud data center operations.

The dataset includes:

Dataset Source: The synthetic dataset used in this study was generated using CloudSim 5.0's workload generator. The dataset parameters were configured to simulate realistic cloud data center operations based on workload patterns described in the study [8].

Dataset Availability: The complete dataset has been uploaded to Kaggle and is publicly accessible for reproducibility: Kaggle Dataset Link: https://www.kaggle.com/datasets/ibiyomimichael/load-balancing-in-cloud-data-centers

Experimental Setup

The hybrid model was implemented in Python and evaluated using CloudSim-based simulation. A synthetic dataset comprising 100,000 data points was generated, capturing CPU, memory, disk, and network metrics across multiple servers and tasks. The dataset was split 80/20 for training and testing.

Test	Servers	Tasks	Rationale
Test 1	10	20	Baseline configuration for performance comparison
Test 2	15	50	Moderate workload to test scalability
Test 3	20	75	High workload to test system limits
Test 4	4	100	Resource-constrained scenario testing efficiency

TABLE 2: Experimental Test Scenarios

Parameter	Test 1	Test 2	Test 3	Test 4
Servers	10	20	25	4
Tasks	25	50	75	100
Ants	15	10	20	50
Iterations	50	100	50	100

TABLE 3: Experimental Setup

Tables 2 and 3 show that four test scenarios were conducted with varying configurations.

Rationale for Test 4: Test 4 uses 4 servers for 100 tasks to simulate resource-constrained scenarios where servers must handle significantly more tasks than the available resources, testing the model's efficiency under high load conditions.

Evaluation Metrics

Performance was evaluated using the following metrics:

Response Time (ms): Average time from task submission to completion

Throughput (tasks/second): Number of tasks processed per second

Resource Utilization (%): Average CPU, memory, and network utilization across servers

Load Imbalance Degree: Standard deviation of load across servers

LSTM Performance: MSE, MAE, RMSE, MAPE, and inference time

Baseline Comparisons

The hybrid model was compared against the following configurations:

Hybrid (Full): Complete LSTM + ACO + Throttle

Without LSTM: ACO + Throttle only (no prediction)

Without ACO: LSTM + Throttle only (random allocation)

Without Throttle: LSTM + ACO only (no dynamic control)

Each configuration was executed 10 times with different random seeds, and average results are reported.

LSTM Prediction Performance

The LSTM model demonstrated strong predictive performance with the following metrics:

Metric	Value

Mean Squared Error (MSE)	0.0836
Mean Absolute Error (MAE)	0.2483
Root Mean Squared Error (RMSE)	0.2892
Mean Absolute Percentage Error (MAPE)	2.17%
Inference Time	0.7249 seconds

TABLE 4: LSTM Prediction Performance

Table *4* shows the results, as the low MAPE of 2.17% indicates that prediction errors average only 2.17% of actual values, confirming high reliability for practical applications. The inference time of 0.7249 seconds supports real-time deployment.

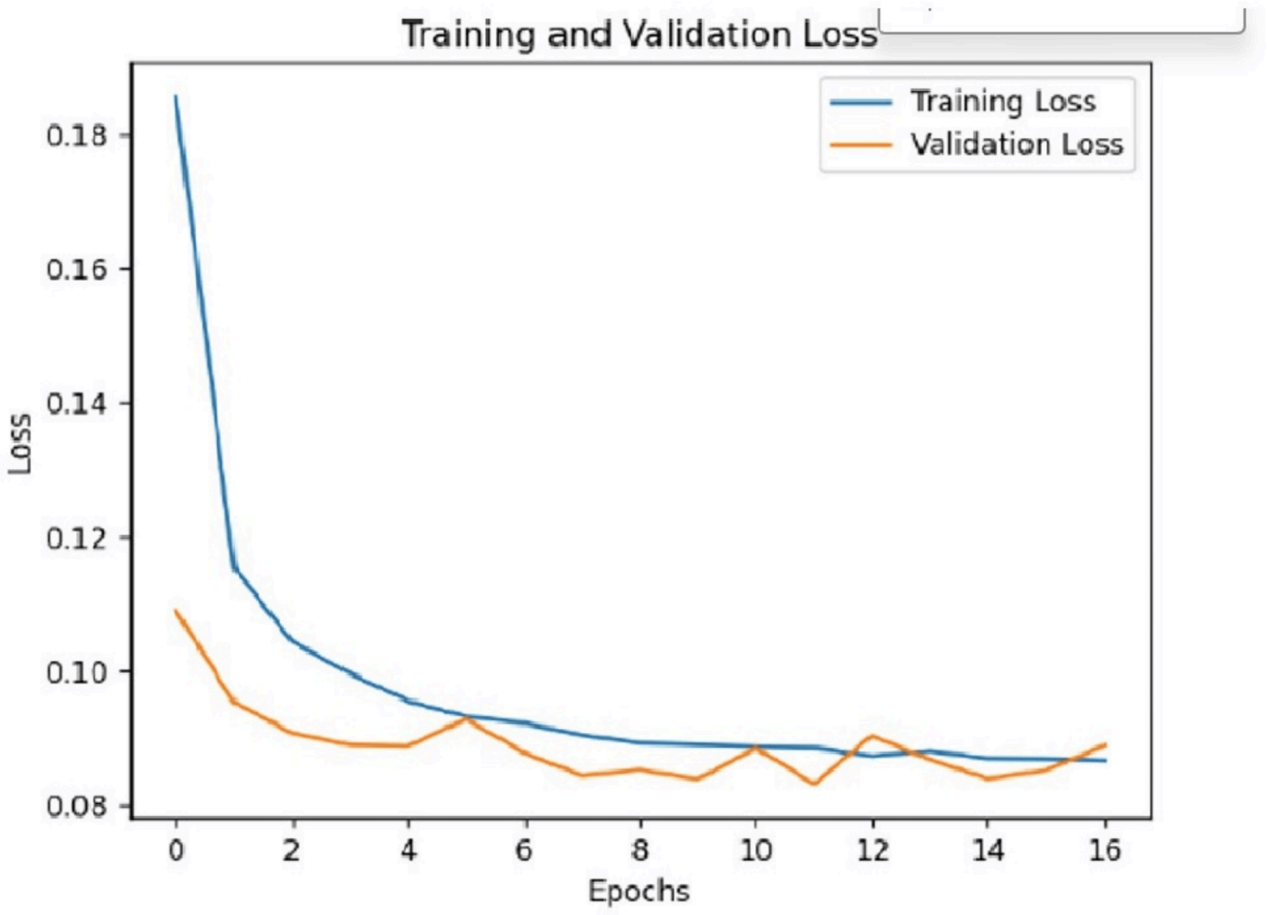


FIGURE 2: Training and Validation Loss Curves

The loss curves demonstrate a consistent decrease over 20 epochs, with both training and validation loss converging without significant divergence, indicating effective learning without over-fitting (Figure *2*).

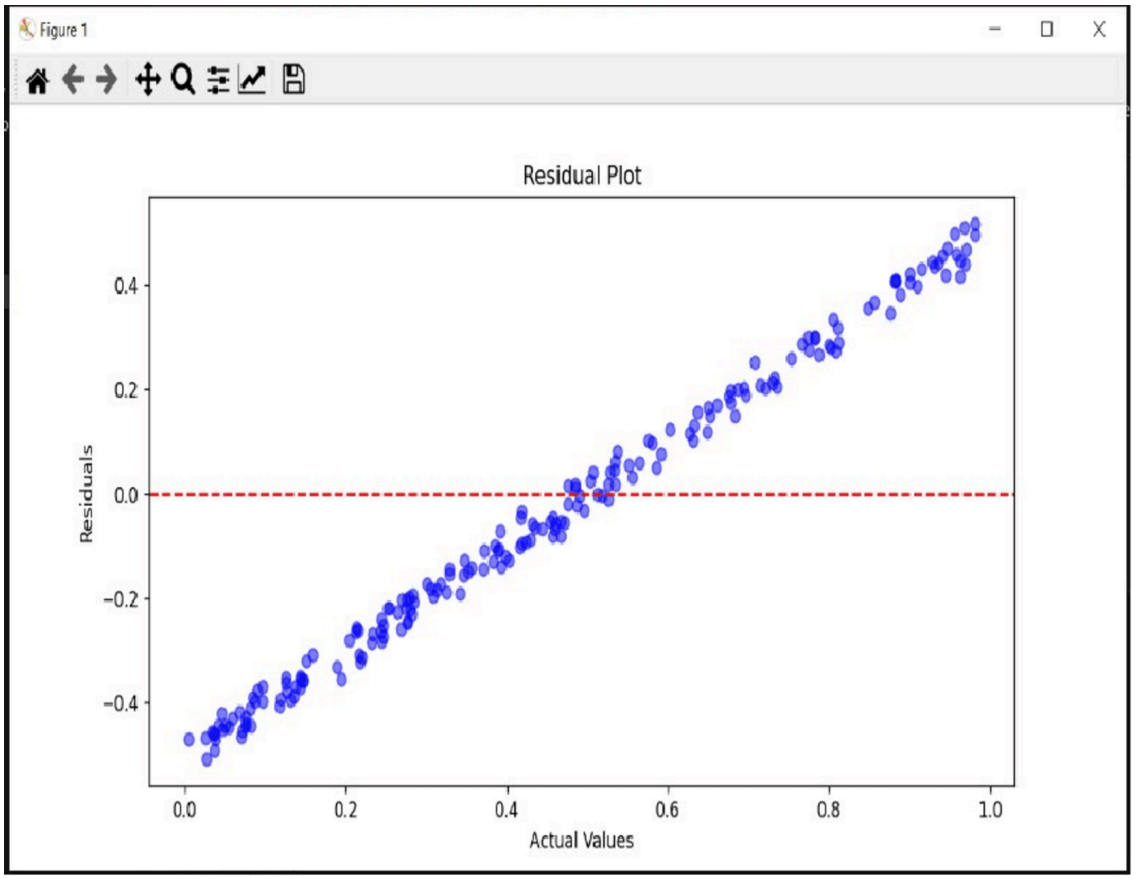


FIGURE 3: Residual Plot

Residual analysis revealed random scatter around the zero line without systematic patterns, confirming that the model captures underlying data relationships without bias (Figure *3*).

Ablation Study

Configuration	Response Time (ms)	Throughput (tasks/second)	Resource Utilization (%)
Without LSTM	180	120	68
Without ACO	210	100	60
Without Throttle	160	130	72
Hybrid (Full)	**120**	**150**	**85**

TABLE 5: Ablated Hybrid Model Configurations

LSTM, Long Short-Term Memory; ACO, Ant Colony Optimization

An ablation study systematically evaluated the contribution of each component (Table 5):

The complete hybrid model achieved:

34.5% faster response time compared to the average of ablated configurations:

Average response time of ablated configurations = (180 + 210 + 160) / 3 = 183.3 ms

Improvement = (183.3 - 120) / 183.3 × 100 = 34.5%

28.6% higher throughput compared to the average of the ablated configurations:

Average throughput of ablated = (120 + 100 + 130) / 3 = 116.7 tasks/second

Improvement = (150 - 116.7) / 116.7 × 100 = 28.6%

27.5% better resource utilization compared to the average of ablated configurations:

Average utilization of ablated configurations = (68 + 60 + 72) / 3 = 66.7%

Improvement = (85 - 66.7) / 66.7 × 100 = 27.5%

The complete hybrid model achieved 34.5% faster response time, 28.6% higher throughput, and 27.5% better resource utilization compared to the average of the ablated configurations.

Performance Analysis

Response Time: The hybrid model achieved a 120 ms response time, outperforming configurations without LSTM (180 ms), without ACO (210 ms), and without Throttle (160 ms). LSTM predictions enable proactive resource allocation, reducing queue waiting times. ACO optimization minimizes task migration overhead, while Throttle control prevents bottleneck formation.

Throughput: At 150 tasks/second, the hybrid model demonstrates superior processing capacity. The combination of predictive scheduling and dynamic control allows the system to maintain high throughput even under variable workloads.

Resource Utilization: Achieving 85% utilization, the hybrid model significantly reduces resource waste compared to traditional approaches. The ACO algorithm's global optimization capabilities distribute workloads efficiently, while Throttle mechanisms prevent both overload and underutilization.

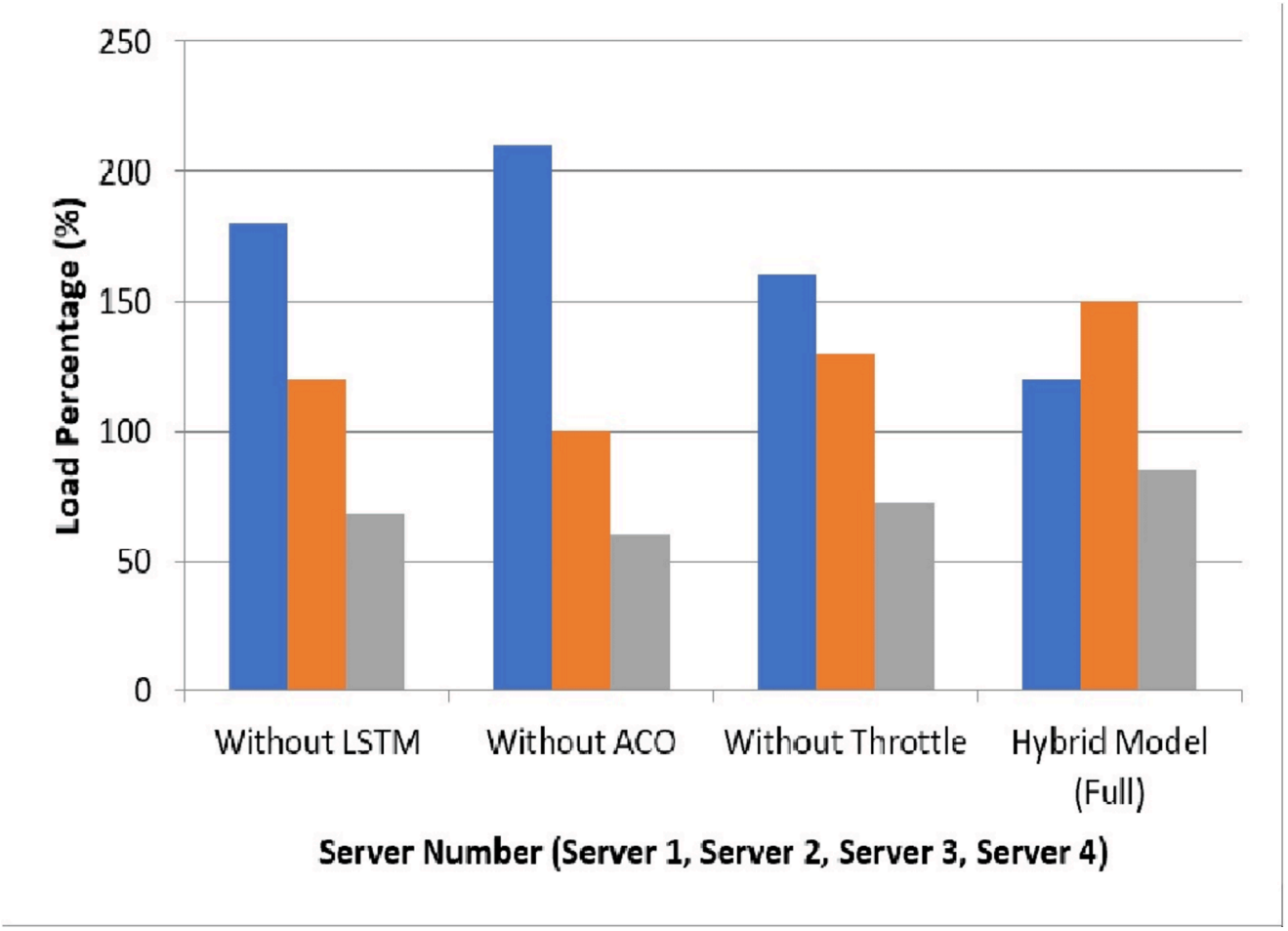


FIGURE 4: Load Distribution Across Servers (Test 4 - 4 Servers, 100 Tasks)

LSTM, Long Short-Term Memory; ACO, Ant Colony Optimization

Visual analysis of load distribution shows balanced allocation across servers, with no single server experiencing disproportionate load, confirming effective load balancing even under resource-constrained conditions (Figure *4*).

Discussion

The results demonstrate that integrating predictive deep learning with metaheuristic optimization and dynamic control mechanisms yields substantial performance improvements. The LSTM model's ability to forecast workload patterns addresses a critical limitation of traditional approaches that rely solely on reactive decisions. By anticipating future loads, the system can pre-allocate resources, reducing response times and preventing overloads.

The ACO algorithm's probabilistic search capabilities complement LSTM predictions by exploring the solution space effectively. Unlike deterministic algorithms that may converge to suboptimal solutions, ACO maintains diversity through pheromone-based exploration, enabling discovery of near-optimal task assignments.

The Throttle algorithm provides essential stability, particularly during workload surges. While LSTM predictions enable anticipation and ACO optimization enables efficient allocation, Throttle control ensures that sudden demand spikes do not destabilize the system.

Compared to existing approaches, the proposed hybrid model offers several advantages. Unlike traditional methods that lack adaptability, the model responds dynamically to changing conditions. Unlike pure metaheuristic approaches that operate without predictive guidance, LSTM-informed ACO makes more informed decisions. Unlike deep learning-only

solutions, the integration of optimization and control mechanisms ensures practical deployability.

Conclusions

This paper presented a hybrid load balancing model integrating LSTM networks, ACO, and Throttle algorithms for cloud computing environments. The proposed approach addresses fundamental limitations of traditional load balancing methods by incorporating predictive capabilities, intelligent optimization, and dynamic control mechanisms. Experimental evaluation demonstrated that the hybrid model achieves superior performance across key metrics: 120 ms response time, 150 tasks/second throughput, and 85% resource utilization. Ablation studies confirmed the synergistic contribution of each component, with the complete model significantly outperforming configurations lacking individual elements. The LSTM predictor achieved 2.17% MAPE with 0.7249-second inference time, confirming suitability for real-time applications. The ACO algorithm successfully optimized task assignments based on predicted loads, while Throttle mechanisms ensured system stability during workload fluctuations.

The study contributes to the body of knowledge in cloud computing by demonstrating the potential of integrating deep learning-based workload prediction with metaheuristic optimization and dynamic load-balancing mechanisms. The proposed integration of LSTM, ACO, and Throttle provides a basis for further research into hybrid intelligent resource-management techniques. The study also provides a controlled experimental framework through which researchers can investigate the effects of workload prediction and optimization on cloud performance metrics such as response time, throughput, resource utilization, and load imbalance. However, because the evaluation is based primarily on synthetic workloads and CloudSim simulation, the findings should be interpreted as a methodological contribution rather than definitive evidence of performance in production cloud environments. For cloud service providers, adopting hybrid load balancing models can yield significant improvements in operational efficiency and user experience. Implementation should prioritize continuous monitoring of workload patterns to maintain accurate LSTM predictions, adaptive parameter tuning for ACO (α, β, ρ) based on environmental characteristics, and integration with existing orchestration frameworks (Kubernetes, OpenStack) for practical deployment.

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

Concept and design: Michael A. Ibiyomi

Acquisition, analysis, or interpretation of data: Michael A. Ibiyomi

Drafting of the manuscript: Michael A. Ibiyomi

Critical review of the manuscript for important intellectual content: Michael A. Ibiyomi

Supervision: Michael A. Ibiyomi

Disclosures

Human subjects: All authors have confirmed that this study did not involve human participants or tissue.

Animal subjects: All authors have confirmed that this study did not involve animal subjects or tissue.

Conflicts of interest: In compliance with the ICMJE uniform disclosure form, all authors declare the following:

- **Payment/services info:** All authors have declared that no financial support was received from any organization for the submitted work.
- **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted

work.

- **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

Data Availability Statements

The datasets (and/or code) supporting this study are available from the corresponding author upon reasonable request. All data generated or analyzed during this study are included in this published article and/or its appendices.

References

1. Goudarzi M, Ilager S, Buyya R: Cloud computing and Internet of Things: Recent trends and directions. New Frontiers in Cloud Computing and Internet of Things. Springer, Switzerland; 2022. 3-29. 10.1007/978-3-031-05528-7_1
2. Gupta MR, Sharma OP: A review exploration of load balancing techniques in cloud computing. Educational Administration: Theory and Practice. 2024, 30:580-590. 10.53555/kuey.v30i2.1600
3. Shafiq DA, Jhanjhi NZ, Abdullah A: Load balancing techniques in cloud computing environment: A review. Journal of King Saud University Computer and Information Sciences. 2022, 34:3910-3933. 10.1016/j.jksuci.2021.02.007
4. Bharadiya JP: Exploring the use of recurrent neural networks for time series forecasting. International Journal of Innovative Science and Research Technology. 2023, 8:2023-2027. 10.5281/zenodo.8002429
5. Kanellopoulos D, Sharma VK: Dynamic load balancing techniques in the IoT: A review. Symmetry. 2022, 14:2554. 10.3390/sym14122554
6. Dorigo M, Stützle T: Ant colony optimization: Overview and recent advances. Handbook of Metaheuristics. International Series in Operations Research & Management Science. Gendreau M, Potvin JY (ed): Springer, Cham; 2019. 272:311-351. 10.1007/978-3-319-91086-4_10
7. Singh SP, Kumar R, Sharma A, Nayyar A: Leveraging energy-efficient load balancing algorithms in fog computing. Concurrency and Computation: Practice and Experience. 2022, 34:e5913. 10.1002/cpe.5913
8. Belgaum MR, Musa S, Alam MM, Mohd Su'ud M: A systematic review of load balancing techniques in software-defined networking. IEEE Access. 2020, 8:98612-98636. 10.1109/access.2020.2995849
9. Kazeem MA, Bamidele JA, Ogundele RO, Misra S, Abidemi EA: Applicability of MMRR load balancing algorithm in cloud computing. International Journal of Computer Mathematics: Computer Systems Theory. 2020, 6:7-20. 10.1080/23799927.2020.1854864
10. Alsadie D: TSMGWO: Optimizing task schedule using multi-objectives grey wolf optimizer for cloud data centers. IEEE Access. 2021, 9:37707-37725. 10.1109/access.2021.3063723
11. Kishor A, Chakarbarty C: Task offloading in fog computing for using smart ant colony optimization. Wireless Personal Communications. 2022, 127:1683-1704. 10.1007/s11277-021-08714-7
12. Malti AN, Hakem M, Benmammar B: A new hybrid multi-objective optimization algorithm for task scheduling in cloud systems. Cluster Computing. 2024, 27:2525-2548. 10.1007/s10586-023-04099-3
13. Simaiya S, Lilhore UK, Sharma YK, et al.: A hybrid cloud load balancing and host utilization prediction method using deep learning and optimization techniques. Scientific Reports. 2024, 14:1337. 10.1038/s41598-024-51466-0
14. Dastagiraiah C, Reddy VK, Rani KP, Veeranna T: Optimizing load balancing using deep learning in a cloud environment. IEEE Xplore In 2024 11th International Conference on Computing for Sustainable Global Development (INDIACom). 2024, 1374-1380. 10.23919/INDIACom61295.2024.10498539
15. Manickavasagan V, Petchiammal E, Sudha SR, Rajesh C, Sree SJ, Yamini VS: Enhancing cloud resource allocation efficiency with Harris hawk optimization algorithm (HHOA). 2024 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), Bhopal, India. 2024, 1-6. 10.1109/SCEECS61402.2024.10481949
16. Ibiyomi MA, Akintola KG, Osuolale FA: A hybrid load balancing model for optimizing performance of cloud data centers. Journal of Scientific and Technological Research. 2025, 7:1-12. 10.37933/nipes/7.4.2025.SI50

17. Load Balancing in Cloud Data Centers. (2025). Accessed: November 10, 2026: https://www.kaggle.com/datasets/ibiyomimichael/load-balancing-in-cloud-data-centers.