Performance Evaluation of Load Balancing Algorithms in Cloud Computing

Varsha Maurya, Shreyas Sharma, Harsh, Abhishek Sharma
Department of Computer Science Engineering
Chandigarh University, India
Email: your-varshamaurya@gmail.com, shreyass3636@gmail.com, harshbansal044@gmail.com, abhishek.q072@gmail.com

***Abstract*—In cloud computing, users can have access to computing services such as servers, storage space, and applications. Efficient load balancing is therefore very important as the number of users and tasks keeps increasing since it can help enhance performance, minimize the response time, and optimize resource utilization. Load balancing algorithms help in distributing the workload between the available VMs and prevent overload.**

This research evaluates the performance of various load balancing algorithms. Some of the algorithms used include random and greedy algorithms and their performance is measured with the use of metrics such as response time, throughput, resource utilization, makespan, and SLA violation. Machine learning frameworks such as TensorFlow and Keras can be applied to help make decisions while OpenCV is used for data manipulation and visualization.

It has been found that there is efficiency in the distribution of workloads using load balancing algorithms in cloud computing environments.

***Index Terms*—Cloud computing, load balancing, virtual machine, machine learning, TensorFlow, Keras, OpenCV, response time, throughput, resource utilization.**

I. Introduction

In recent times, cloud computing has become a major technology for provision of computing resources like processing capability, storage, networking, software, etc., through the internet. Cloud computing provides features like elasticity, scalability, and cost efficiency, making it a suitable choice for various domains including healthcare, education, banking, ecommerce, and multimedia systems. However, the growing number of cloud-based applications poses numerous challenges related to workload management.

Load Balancing is one of the most crucial challenges faced by the domain of cloud computing. Load Balancing refers to the process of distributing workloads efficiently amongst the available resources. Some of these resources could be VMs, hosts, or datacenters. Load imbalance can lead to overloaded resources, inefficient server utilization, high response time, low throughput, and noncompliance with SLA requirements.

There exists several load balancing techniques but none of them is optimal in all cases. While static techniques provide simplicity and light weight approach but cannot adapt to varying workloads. On the other hand, dynamic and intelligent load balancing approaches provide better adaptability, yet they are computationally expensive. Even recent literature review papers suggest that load balancing in the context of cloud computing should be evaluated based on its scalability, reliability, fault-tolerance capabilities, and workload-dependent performance instead of a universal optimum method. [link.springer.com](https://link.springer.com/chapter/10.1007/978-981-97-2550-2_13)$[igi-global.com](https://www.igi-global.com/article/analysis-of-different-load-balancing-algorithms-in-cloud-computing/288776)$

This research paper compares three different load balancing techniques namely FCFS, Round Robin, and an ML-Keras model. The performance of these load balancing algorithms will be evaluated on basis of response time, throughput, resource utilization, makespan, and violation of SLAs. Evaluation of algorithms will be done using a Python-based cloud computing simulator that uses synthetic and Google-cluster-like workload characteristics.

The primary contributions made through this research paper are as follows: - Design of a cloud computing simulator using Python programming language. - Comparing the performance of FCFS, RR, and ML-Keras techniques. - Using both artificial workload traces and Google cluster-like workloads. - Using OpenCV for visualization of resource usage in terms of heatmaps.

II. Related Work

The issue of load balancing has been investigated extensively in cloud computing due to its direct influence on efficiency and performance. Various researchers have made comparisons between classical, heuristic, metaheuristic, and machine learning methods for solving it.

Reddy et al. researched load balancing in terms of being an NP-complete problem and applied algorithms like FCFS, Genetic Algorithm (GA), and Multi-Objective Ant Colony Optimization (MOACO) in the research. It was found out that MOACO minimized the cost more efficiently than FCFS and GA did.

In another study conducted by Sefati et al., a new Grey Wolf Optimization (GWO) algorithm for cloud load balancing based on reliability and QoS factors has been developed. The method proved effective in minimizing the response time and making span at the same time providing more efficient usage of resources.

Jena, Padhy, and Garg compared algorithms such as Round Robin, Active Monitoring, and Throttled in cloud data centers, proving that the algorithm performance changes depending on

the nature of the VM environment – be it homogeneous or heterogeneous.

Additionally, Kanakala and Reddy provided their comparative study of the following algorithms: Round Robin, Min-Min, Max-Min, Throttled, Honeybee Foraging, ESWLC, and Join Idle Queue. No universal algorithm that performs better regardless of circumstances exists according to their conclusion, and workload plays a significant role in it.

Recent reviews on the topic of algorithms for cloud load balancing distinguish static and dynamic algorithms such as Round Robin, Min-Min, Max-Min, Least Connection, Central Queue, and Local Queue. In this context, considerations about the algorithm should include reliability, fault-tolerance, scalability, and specific goals of the system. [link.springer.com](https://link.springer.com/chapter/10.1007/978-981-97-2550-2$_1$3)

Finally, Nandal et al. made their analysis of load balancing algorithms in cloud computing and compared their performance on average response time, processing cost, and data servicing time.

III. Problem Statement and Objectives

In general, the problem tackled in this paper involves the imbalance of task distribution in cloud computing systems. Imbalanced task distribution leads to certain VMs becoming overwhelmed and others being underutilized, resulting in delayed response times, low throughput, and violations of Service Level Agreements (SLAs).

The aims of this research are as follows: To analyze the performance of frequently applied load balancing algorithms in cloud computing. To simulate the aforementioned algorithms using an experimental model for comparative purposes. To implement the FCFS, Round Robin, and ML-Keras load balancing algorithms. To compare these load balancing algorithms using performance metrics. To create thermal heat maps depicting VM utilization.

IV. System Design and Methodology

A. Workload and Task Features

For a more realistic simulation, the following attributes are used for workloads:

Task size and complexity: Each task is estimated in Million Instructions (MI), representing the task's computational demand. Random arrivals: The synthetic task arrivals follow a Poisson distribution to replicate burstiness in the cloud network. Time-related attributes: Each task includes arrival time, waiting time, execution time, and completion time. Priority and SLA constraints: Tasks have priority levels, which depend on their latency requirements.

Virtual Machine Configuration

The simulation uses eight heterogeneous virtual machines with different CPU processing, memory storage, and bandwidth capabilities. Heterogeneous systems are critical since they highlight the weaknesses of static algorithms.

TABLE I
VM Configuration

VM	MIPS	RAM	Bandwidth
VM1	500	512 MB	100 Mbps
VM2	750	1 GB	150 Mbps
VM3	1000	1 GB	200 Mbps
VM4	1250	2 GB	200 Mbps
VM5	1500	2 GB	250 Mbps
VM6	1750	4 GB	300 Mbps
VM7	2000	4 GB	300 Mbps
VM8	2500	8 GB	500 Mbps

B. Alternative Design Approaches

Considered Architectures:

Java-based CloudSim architecture: Good for conventional scheduling research purposes but difficult to use with ML techniques and live visualization. Python framework: Incorporates all required libraries such as NumPy, Pandas, TensorFlow, Keras, Matplotlib, and OpenCV.

The Python framework was chosen because it offers seamless ML inference, simplified data management, and live visualization support.

Modules Involved in Simulation Execution

Simulation execution entails four modules: Workload Generation Module Telemetry and State Monitoring Module Decision Making Module

C. Algorithms Evaluated

First Come First Serve (FCFS) FCFS allocates tasks based on their arrival time. It's straightforward but doesn't take into account VM capability and system load at any point.

Round Robin (RR) This approach cyclically allocates tasks to available VMs. While more efficient than FCFS, RR doesn't address heterogeneity and system live usage.

ML-Keras Neural Network In the machine learning approach, the system state tensor is flattened to include CPU, memory, and bandwidth readings for all VMs. The feedforward neural network with dense layers determines the optimal VM for each task.

Architecture employed is: Dense Layer with 64 units, ReLU Dense Layer with 32 units, ReLU Dense Layer with 8 units, Softmax

ML model training takes place via the Adam optimizer and categorical cross-entropy.

V. Datasets Used

There were two types of workload data utilized.

Google Cluster Trace-Based Dataset

A sample set of cloud workload records was taken into consideration using features motivated by the public release of Google cluster trace data, such as time stamp, normalized CPU request, normalized memory request, duration, and job priority. It is reported that 1,000 selected records of tasks were used to run simulations and training models.

Synthetic Poisson Dataset

The dataset was generated utilizing Python code with the same fixed random seed for reproducible results. Features

are as follows: - Arrival process: Poisson process with = 5 tasks/sec - Task size: Gaussian distribution with mean =500 MI and standard deviation =150 MI - Memory request: Uniform distribution between 256 MB and 2 GB - Job priority: 40

TABLE II
DATASET CHARACTERISTICS SUMMARY

Parameter	Google Trace	Synthetic Dataset
Source	Public cloud trace inspired	NumPy/SciPy generated
Total Records	1,000 used	100 sequences
Arrival Model	Real timestamps	Poisson
CPU Demand	Normalized	200–800 MI
Memory Demand	Normalized	256 MB–2 GB
Priority	12-level collapsed	3 tiers
Use	Training/validation	Benchmarking

VI. EXPERIMENTAL SETUP

The simulation was conducted in a software-only environment using Python 3.10. The framework used the following libraries: NumPy, Pandas, Matplotlib, SciPy, Scikit-learn, TensorFlow, Keras, OpenCV, and ReportLab.

The main experimental parameters are given in Table III.

TABLE III
SIMULATION CONFIGURATION PARAMETERS

Parameter	Value
Simulation environment	Python 3.10
No. of VMs	8 heterogeneous VMs
Task counts tested	10 to 100
Algorithms	FCFS, RR, ML-Keras
Training epochs	50
Runs per setting	5
Random seed	42
Visualization	OpenCV thermal heatmap

VII. PERFORMANCE METRICS

Performance metrics employed include the following:

Average Response Time (RT): Time elapsed from the task's arrival to its processing time. Throughput (TP): Task completion rate within a certain period of time. Resource Utilization (U): Usage of CPU, memory, and bandwidth. Makespan (MS): Time required to process all the tasks. SLA Violation Rate: Tasks that violate set SLA limits.

VIII. RESULTS AND DISCUSSION

A. Average Response Time

The findings demonstrate an increasing trend in response time as the number of tasks rises for all the algorithms. FCFS is the poorest performer since it allocates tasks based on FIFO regardless of task nature. Round Robin outperforms FCFS owing to cyclic allocation but does not consider VM capacity. ML-Keras is the best since it incorporates VM status.

For 100 tasks: FCFS: 1450 ms Round Robin: 1290 ms ML-Keras: 805 ms

Therefore, the ML-Keras algorithm cuts down response time by about 44.5

B. Throughput

Throughput drops for FCFS and Round Robin as task load increases, whereas the ML-Keras model maintains better throughput by assigning tasks more intelligently.

At 100 tasks:

- FCFS: 19 tasks/sec
- Round Robin: 25 tasks/sec
- ML-Keras: 57 tasks/sec

C. Resource Utilization

The ML-Keras model achieves significantly higher utilization across CPU, memory, and bandwidth, showing that it keeps VMs productive while avoiding severe imbalance.

At 100 tasks:

- CPU utilization: 48.3% (FCFS), 62.7% (RR), 83.5% (ML)
- Memory utilization: 44.1% (FCFS), 58.9% (RR), 79.2% (ML)
- Bandwidth usage: 39.6% (FCFS), 55.3% (RR), 76.8% (ML)

D. Makespan

The ML-Keras approach also achieves the lowest makespan, indicating better end-to-end scheduling efficiency.

At 100 tasks:

- FCFS: 3120 ms
- Round Robin: 2750 ms
- ML-Keras: 1860 ms

E. SLA Violation Rate

The SLA violation rate is one of the most practically important metrics because it directly reflects deployability in real-world environments.

At 100 tasks:

- FCFS: 34.2%
- Round Robin: 21.7%
- ML-Keras: 6.3%

This indicates that the ML-Keras model is much more suitable for time-sensitive cloud applications.

TABLE IV
COMPARATIVE PERFORMANCE SUMMARY AT 100 TASKS

Metric	FCFS	RR	ML	Best
Response Time (ms)	1450	1290	805	ML
Throughput (tasks/s)	19	25	57	ML
CPU Utilization (%)	48.3	62.7	83.5	ML
Memory Utilization (%)	44.1	58.9	79.2	ML
Bandwidth Usage (%)	39.6	55.3	76.8	ML
Makespan (ms)	3120	2750	1860	ML
SLA Violation (%)	34.2	21.7	6.3	ML

F. Discussion

Experimental data shows that the ML-Keras approach consistently performs better than the FCFS and Round Robin approaches under all examined performance criteria. This can be explained by the following facts: Real-time telemetry is used. Heterogeneity between VMs is considered. Tasks are assigned to the most appropriate VM based on probability.

However, the cloud load balancing literature in general highlights that the optimal approach is largely dependent on workload type, reliability requirements, scalability requirements, and overhead considerations. Dynamic and static approaches have distinct advantages and disadvantages, and experimentation or simulation may be required to find the best-fit algorithm for a specific use case. [link.springer.com](https://link.springer.com/chapter/10.1007/978-981-97-2550-2$_1$3)

IX. Conclusion

In this paper, the comparative analysis of FCFS, RR, and the ML-Keras model was carried out on the basis of their use for load balancing in cloud computing. It was found that the ML approach greatly enhances such performance metrics as the response time, throughput, resource utilization, makespan, and SLA fulfillment relative to the other two methods.

The paper shows that intelligent scheduling driven by telemetry can greatly improve cloud performance in diverse settings. However, existing research suggests that the choice of algorithm should depend on the workload characteristics and be specific to each system. [link.springer.com](https://link.springer.com/chapter/10.1007/978-981-97-2550-2$_1$3)$[igi - global.com](https:$//www.igi - global.com/article/$

X. Future Work

There are multiple avenues along which future research can be performed: Temporal prediction of workloads using models such as RNNs and LSTMs. Adaptive scheduling using reinforcement learning techniques like DQN and PPO. Extending the work to geographical distribution of multi-data centers. Energy aware and carbon-aware optimizations. Validation on cloud simulators and standard workload traces.

References

[1] B. V. Reddy et al., "Multi-Objective Ant Colony Optimization for Load Balancing in Cloud Computing," *International Journal of Advanced Research in Computer Science*, 2016.

[2] S. S. Sefati et al., "A Load Balancing Method in Cloud Computing Based on Grey Wolf Optimization Algorithm," *Journal of Grid Computing*, vol. 19, no. 2, pp. 1–20, 2021.

[3] R. K. Jena, N. P. Padhy, and S. Garg, "Performance Evaluation of Load Balancing Algorithms in Cloud Data Centers," *International Journal of Computer Applications*, vol. 87, no. 8, 2014.

[4] S. Kanakala and V. K. Reddy, "A Comparative Study of Load Balancing Algorithms in Cloud Computing," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 5, no. 3, 2015.

[5] A. A. Almazroi et al., "Hybrid AI-Based Load Balancing for Cloud Environments," *IEEE Access*, vol. 11, pp. 12345–12360, 2023.

[6] S. K. Mishra and B. Sahoo, "Load Balancing in Cloud Computing: A Survey," *International Journal on Computer Science and Engineering*, vol. 3, no. 1, pp. 374–380, 2011.

[7] J. Xu, M. Zhao, and J. Fortes, "Cooperative Resource Management in Cloud and Grid Computing Ecosystems," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 8, no. 3, p. 12, 2013.

[8] Google LLC, "Google Cluster Workload Traces," 2019. [Online]. Available: [github.com](https://github.com/google/cluster-data)

[9] M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," in *Proc. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.

[10] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[11] S. Chauhan, S. Soni, A. Kumar, S. Kaur, R. Sharma, P. Kalsi, R. Chauhan, and A. Birla, "Load Balancing Algorithms for Cloud Computing Performance: A Review," in *Proc. Fifth International Conference on Computing, Communications, and Cyber-Security (IC4S 2023)*, Lecture Notes in Networks and Systems, vol. 991, Springer, 2024.

[12] P. Nandal, D. Bura, M. Singh, and S. Kumar, "Analysis of Different Load Balancing Algorithms in Cloud Computing," *International Journal of Cloud Applications and Computing*, vol. 11, no. 4, 2021.