International Journal of Computer Applications (0975 - 8887)
Volume 187 - No.133, August 2026

Evaluation and Combining of Load-balancing AI-driven Models, including IaCloud Model

Anouar Ben Halima
Mohammed V University in Rabat
Rabat, Morocco
https://orcid.org/0009-0009-1537-0462

Hafssa Benaboud
Mohammed V University in Rabat
Rabat, Morocco
https://orcid.org/0000-0002-5027-8217

ABSTRACT

Large Language Models (LLMs) have recently achieved significant progress in natural language processing tasks, including question answering and assisting users, such as in cloud computing. Cloud computing is a composite of various fields, including LLMs, which is a straightforward approach to enhancing the performance of the entire cloud. Therefore, AI-driven load balancing has been developed in cloud computing for a long period to benefit from machine learning to enhance the performance of cloud computing. This study presents a comparative evaluation of several state-of-the-art LLMs, including OpenAI GPT-4 and Google Gemini, with our proposed model (IaCloud[1]) in predicting the most appropriate load-balancing techniques in the cloud environment.

General Terms

Cloud computing, Load balancing, Artificial Intelligence

Keywords

Cloud computing, Load balancing, LLMs, agentic AI, AI-driven, IaCloud

1. INTRODUCTION

Cloud computing is a modern technology that individuals, companies, and organizations widely use to store data and run applications, ensuring a high level of access, performance, and security. It provides a set of shared resource pools, such as data center servers or routers. Additionally, numerous integrated solutions have been developed that enhance the cloud computing environment using machine learning and large language models (LLMs) across different levels, including SaaS, PaaS, IaaS, and LbaaS (load balancing as a service) [1, 2]. Large language models are advanced artificial intelligence systems designed to understand and generate human language. LLMs are widely used in modern applications, including virtual assistants, chatbots, education, and research. Machine learning is a branch of artificial intelligence where computers learn patterns from data and improve the LLM's performance without being explicitly programmed.

[1]Patent MA67595A1

The collaboration of machine learning and LLMs creates flexible AI-driven systems that are autonomous and AI-driven cloud computing. Likewise, researchers endeavor to optimize cloud computing services, therefore [1, 3, 4, 5]. Similarly, we will use customized LLMs that aim to select an appropriate load balancing in order to compare it with other common models, such as ChatGPT and Gemini.

As aforementioned, LbaaS is the service responsible for routing and distributing incoming traffic across multiple resources in cloud computing to prevent overloaded or underloaded resources, including both software and hardware components. Load balancing implements various types of algorithms and techniques, both static and dynamic, which also include a diversity of constraints and metrics that subsequently add complexity to selecting a suitable option.

To this end, this model used for this evaluation was created based on prior research from previous studies and then translated into insightful knowledge to satisfy consumers' aims, including important information on load-balancing techniques, strategies, and algorithms from multiple scientific resources.

Quality of Service (QoS) recommends various criteria, known as "metrics selection," used for this evaluation. These criteria are called "metric selection," defined as follows:

(1) Performance: Represents the overall effectiveness of the cloud system after applying a load-balancing algorithm. Higher performance indicates better satisfaction of QoS requirements.

(2) Throughput: The total number of tasks completed within a given period.

(3) Associated Overhead: the additional computational and communication costs incurred during the execution of the load balancing algorithm.

(4) Fault Tolerance: The ability of a load balancing algorithm to maintain correct operation and service availability despite failures of nodes, virtual machines, or network components.

(5) Migration Time: The time required to transfer a task, process, or virtual machine from one resource to another.

(6) Response Time: The delay between a user sending a request and the system's response. Effective load balancing aims to minimize response time.
(7) Resource Utilization: Measures how efficiently cloud resources such as CPU, memory, and bandwidth are used. Higher utilization indicates better resource management.
(8) Scalability: Represents the ability of the system to maintain performance as the workload or number of resources increases.
(9) Power Saving or Energy Consumption: Measures the effectiveness of the load balancing algorithm in reducing energy consumption while maintaining acceptable QoS levels.

During this evaluation, we test the combination of these metrics as inputs while awaiting the models' responses, focusing on the best algorithms and accuracy, including those from ChatGPT and Gemini. Subsequently, we interact with the aforementioned models, relying on prompt engineering by choosing specific prompts for the evaluation. Prompt engineering is the art and practice of designing clear, specific, and structured instructions to guide AI models.

This study presents a comparative study of our customized model against different LLMs, such as ChatGPT and Gemini. Therefore, an evaluation is conducted for the proposed model with different scenarios of metrics. The resulting load balancing recommendations are then compared with those generated by ChatGPT and Gemini.

This paper is organized as follows. Section 2 cites related work. Section 3 presents the methodology. Section 4 displays the outcomes of the evaluation model compared with other known models. Section 5 aims to discuss the results. In the last section 6, we conclude and discuss future research.

2. RELATED WORK

Task scheduling and resource management remain fundamental challenges in cloud computing due to the dynamic nature of workloads, heterogeneous resources, and the need to simultaneously optimize multiple QoS metrics such as response time, cost, throughput, and resource utilization [6]. Traditional scheduling approaches, including heuristic and meta-heuristic algorithms, have been extensively investigated to improve cloud performance. The authors of article [7] reviewed several hybrid meta-heuristic scheduling techniques and highlighted their effectiveness in optimizing resource allocation and execution efficiency in cloud environments. However, these approaches often suffer from limited adaptability in highly dynamic cloud infrastructures and require extensive parameter tuning. Recent advances in artificial intelligence, particularly large language models, have introduced new opportunities for intelligent cloud resource management, particularly when combining multiple tools [8]. [4] proposed an LLM-based cost-aware task scheduling framework for cloud computing systems. Their approach combines Deep Reinforcement Learning (DRL) with LLM reasoning capabilities, where DRL agents generate high-quality scheduling decisions that are subsequently used to fine-tune an LLM acting as a high-level scheduling agent. The proposed framework aims to optimize multiple objectives simultaneously, including average response time, task success rate, and rental cost. Experimental results demonstrated improved generalization capabilities across heterogeneous cloud environments compared with traditional DRL-based schedulers. Nevertheless, the study primarily focuses on cloud task scheduling and does not extensively investigate distributed edge-cloud deployment scenarios. To address the increasing computational demands of LLM inference, Pozi and Sato [5] introduced a data-augmented model routing framework for efficient LLM deployment in edge-cloud environments. Their work employs a supervised routing mechanism that classifies incoming prompts according to difficulty levels and dynamically routes them either to lightweight edge-based models or more powerful cloud-hosted models. Furthermore, the authors proposed a mask-based data augmentation strategy to improve routing accuracy and reduce unnecessary computational overhead. Experimental evaluations showed significant improvements in routing efficiency while maintaining inference accuracy and reducing operational costs. However, the proposed framework mainly focuses on program generation tasks and LLM deployment optimization rather than broader cloud resource scheduling problems. Several recent studies have further explored hybrid cloud-edge architectures for efficient LLM deployment. [9] proposed a multi-objective routing algorithm based on NSGA-II to distribute inference requests among heterogeneous cloud and edge LLM instances. Their approach simultaneously considers response quality, latency, and inference cost, demonstrating substantial cost reductions while maintaining acceptable performance levels. Similarly, [10] developed EACO-RAG, a distributed edge-assisted Retrieval-Augmented Generation (RAG) framework that leverages collaborative edge nodes and adaptive knowledge updates to improve scalability and reduce deployment costs in hybrid environments. These studies emphasize the growing importance of intelligent routing and adaptive workload distribution for large-scale LLM services. In addition, reliability-aware scheduling has attracted considerable attention in multi-cloud environments. Tekawade and Banerjee [11] proposed a cost-effective and reliability-aware scheduler for workflow execution in heterogeneous multi-cloud systems. Their method incorporates pricing mechanisms, reliability constraints, and task dependencies to improve scheduling efficiency. Although effective, the proposed solution relies on traditional optimization techniques and does not exploit the reasoning and decision-making capabilities offered by modern LLMs. Despite the significant progress achieved by existing studies, several research gaps remain. Most traditional scheduling approaches lack adaptability to rapidly changing cloud conditions, while many recent LLM-based solutions focus either on scheduling decisions or inference routing independently. Consequently, there is still a need for adaptive frameworks that integrate LLM reasoning, intelligent routing, and dynamic resource allocation mechanisms to simultaneously optimize performance, cost efficiency, and scalability across cloud and edge-cloud computing environments.

3. METHODOLOGY

The proposed methodology aims to evaluate the effectiveness of a customized artificial intelligence model for recommending the most appropriate load balancing algorithm according to application-specific Quality of Service (QoS) requirements. In the first phase, the customized model was collected from prior research on load balancing in cloud computing, and then translated into insights to satisfy consumers' needs using a decision tree. Initially, we collected important information on load-balancing techniques, strategies, and algorithms from multiple scientific sources, including metrics such as throughput, response time, and tolerance analyses, as well as common static and dynamic techniques of load balancing. The collected knowledge was then

transformed into a structured dataset describing the suitability of each load balancing algorithm for different QoS requirements. Each dataset instance represents a unique combination of the selected QoS criteria, while the corresponding class label identifies the load balancing algorithm considered the most appropriate according to the collected expert knowledge. In the second phase, multiple training and cross-validation data partitions were evaluated to assess the model's generalization capability. Additionally, resampling techniques were employed to estimate the robustness and stability of the model's performance across different data distributions. As a result, the information acquired describes a well-organized collection of load-balancing strategies, including the crucial metric values needed to select the appropriate load-balancing method. The pre-built model was then exposed as a RESTful API that receives QoS requirements as input and returns the recommended load-balancing algorithm together with the associated prediction confidence.

To establish a comparative evaluation, the same 512 QoS scenarios were submitted to three different recommendation approaches: the proposed decision tree model, OpenAI GPT, and Google Gemini. For the large language models, a standardized prompt was designed to ensure that identical information was provided to each model for every experiment. Each prompt contained the same QoS criteria and requested the recommendation of the most suitable load balancing algorithm from the predefined candidate list. This procedure ensured a fair comparison by eliminating variations due to prompt formulation. Otherwise, in our proposed model, it interacts with the HTTPS protocol via the command line, even using curl or another tool.

Ultimately, the prediction results generated by the three approaches were analyzed using descriptive and comparative statistical techniques. Frequency distributions, accuracy measurements, agreement matrices, prediction diversity analysis, and correlation studies were employed to compare the consistency of algorithm selection among the evaluated models. Thereby using statistical insights to help us understand the data, identify patterns, and communicate results more effectively.

4. RESULTS

The experimental evaluation was conducted on 512 unique combinations of QoS requirements. Three approaches were compared: OpenAI GPT, Google Gemini, and the proposed decision tree-based adaptive load balancing model. The analysis revealed clear differences in the algorithm selection patterns across the evaluated approaches. The experiment offers 32 predictable algorithms, such as Round Robin [12], Ant Colony [13], Dynamic Round Robin [12], and FAMLB [14] [15].

The outcomes display that Gemini has a strong preference for PALB [16], selecting it in 167 scenarios, followed by CARTON [17] in 100 scenarios and FAMLB [14] in 53 scenarios. Similarly, GPT favored PALB, which was selected in 242 scenarios, while JoinIdleQueue [18] and FAMLB were selected in 133 and 96 scenarios, respectively. The proposed model offers more space for preferred algorithms and more recommendations, including 15 algorithms. For example, PALB was frequently selected for 32 scenarios, whereas Round Robin was associated with 15 scenarios. Similarly, the Throttled and LBVS algorithms were also recommended for 15 scenarios. These figures, 1, 2, and 3, present

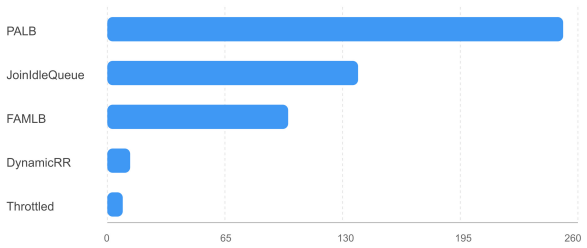

Fig. 1: Top selected algorithms in GPT.

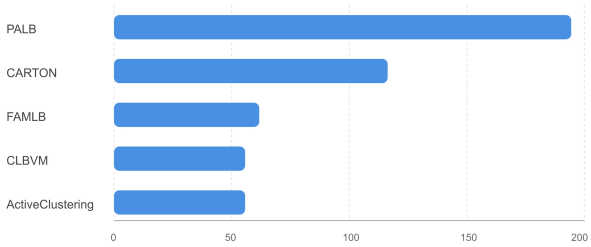

Fig. 2: Top selected algorithms in Gemini.

the aforementioned results.

Generally, the experiment demonstrates that among the 32 evaluated load balancing algorithms, 16 emerged as the most frequently selected. These algorithms can be broadly categorized into two groups based on their selection frequency: a high-priority group, which was consistently recommended across many QoS scenarios, and a lower-priority group, which was selected less frequently but remained suitable for specific QoS requirements.

Additionally, traditional algorithms like Round Robin [12] have low frequency levels, citing ShortestJobSheduling [19] and BiasedRandomSapmling [20] algorithms. Moreover, GPT recommended only 12 of the 32 available load balancing algorithms, leaving 20 algorithms unselected throughout the 512 evaluation scenarios. Gemini demonstrated greater diversity by selecting 17 algorithms while excluding the remaining 15. In contrast, the proposed IaCloud model exhibited the broadest coverage, recommending 23 distinct algorithms and leaving only 9 algorithms unselected: Max-Min [21], HoneyBeeForaging [22], GenetecAlgorithm [23], StochasticHillClimbingTech [24], TaskShedulingbasedOnLB and ACCLB [25], CAB and LFCB [26], VD [27] and GLBS [28]. Table 1 displays a general overview.

Table 1. : General overview of selected algorithms in various models

Metric	GPT	Gemini	IaCloud Model
Total Scenarios	512	512	512
Distinct Algorithms Selected	12	17	23
Diversity Percentage (%)	37.50	53.13	71.88

The agreement analysis (Figure 4) demonstrated that GPT and Gemini produced identical recommendations in 39.26 percent of

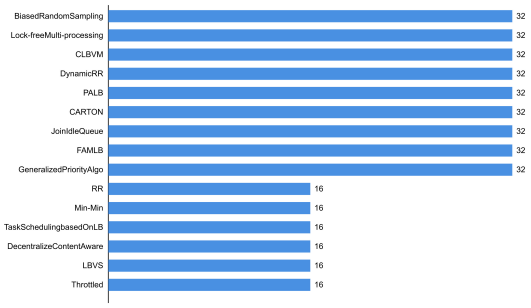

Fig. 3: Frequency of selected algorithms in the decision tree model.

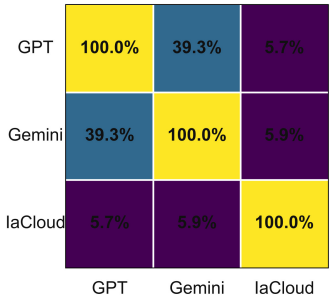

Fig. 4: Agreement matrix for the models.

the evaluated scenarios. In contrast, the agreement between GPT and the proposed model was 5.66 percent, while the agreement between Gemini and the proposed model was 5.86 percent, as shown in this figure.

The confidence analysis further revealed that Gemini produced highly stable predictions with an average confidence of approximately 95 percent. The proposed model exhibited a more balanced distribution of algorithm selections across the available load balancing techniques, indicating its ability to consider a broader range of decision criteria rather than converging on a small subset of frequently selected algorithms.

Figure 5 presents a clustered heatmap illustrating the relationship between QoS criteria and the algorithms selected by the proposed decision tree model. Hierarchical clustering groups algorithms exhibiting similar decision patterns. The results reveal several algorithm families characterized by common QoS objectives. For example, PALB [16], FAMLB [14], and DynamicRR [12] are strongly associated with throughput and response time requirements, whereas Throttled [29] and DecentralizeContentAware [30] are more closely linked to scalability and resource utilization objectives. For example, PALB [16] is strongly associated with throughput, tolerance, and response time but less associated with performance. Another example, DynamicRR [12], is strongly associated with throughput and overhead. This figure represents all of the other options.

The study also offers the clustered version that automatically groups algorithms with similar QoS-selection behavior for the model built by a decision tree. This makes patterns much easier to identify. For instance, cluster one includes DynamicRR [12], GeneralizedPriorityAlgo [20], FAMLB [14], and PALB [16], which appear together because they are strongly associated with the same criteria: throughput, response time, and resource utilization. Cluster 2: CLBVM [31], Lock-free Multi-processing [32], BiasedRandomSampling [20], CARTON [17], Min-Min [33], RR[12]. Performance requirements, strong throughput requirements, and less emphasis on tolerance. As an instance, Cluster 3: Throttled, Decentralized Content-Aware Algorithm [30]. These are highly associated with resource utilization and scalability.

Figure 6. QoS-to-algorithm relationship graph derived from the proposed adaptive decision tree model. The figure illustrates the strongest associations between quality-of-service requirements and the load-balancing algorithms selected by the model. PALB [16] is primarily associated with power-saving and tolerance requirements, whereas DynamicRR [12] is strongly linked to response-time optimization. Throttled exhibits strong associations with scalability and resource utilization objectives, while FAMLB [14] is mainly selected when migration-time considerations are important.

5. DISCUSSION

The results suggest that both LLMs tend to prioritize a limited subset of load balancing strategies when simultaneously satisfying multiple QoS objectives. Conversely, our model uses the entire algorithm space with acceptable accuracy and diversity, offering multiple options. This indicates a broader exploration of the load balancing solution space and a higher degree of adaptation to varying QoS requirements. In addition, the results of agreement analysis, as mentioned, suggest that the proposed decision tree model follows a substantially different decision-making process and is not merely reproducing the recommendations generated by large language models. Overall, the results demonstrate that the proposed adaptive load balancing framework provides a distinct and diversified decision strategy while maintaining competitive performance across a wide range of QoS requirements. It displays the relationship between key criteria and load balancing algorithms using statistical insights.

Furthermore, the clustered graphs demonstrate the variety of the selected models by the proposed method. The first cluster suggests algorithms belong to a family of performance-oriented load-balancing techniques. The second cluster indicates a focus on execution efficiency rather than fault handling. The last cluster

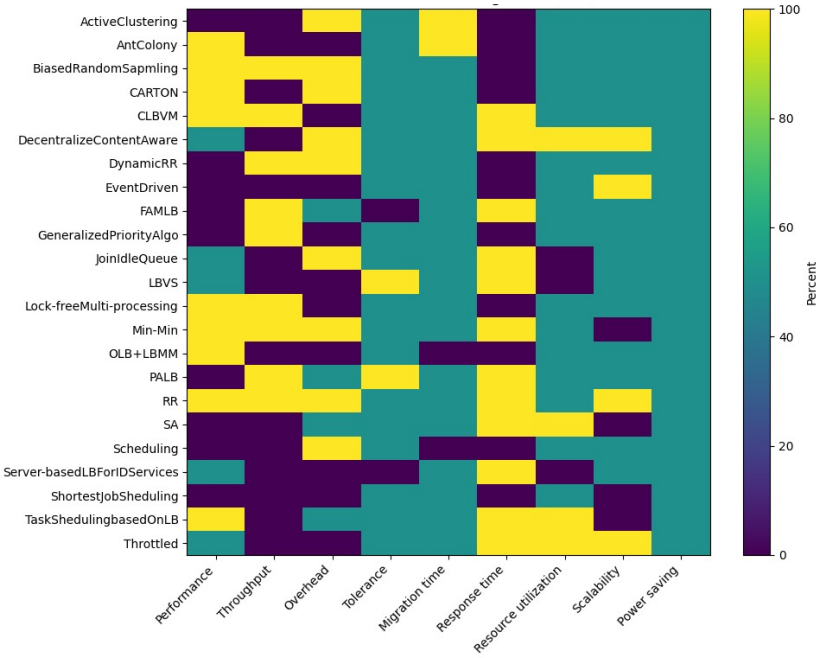

Fig. 5: Clustering of QoS criteria against selected algorithms.

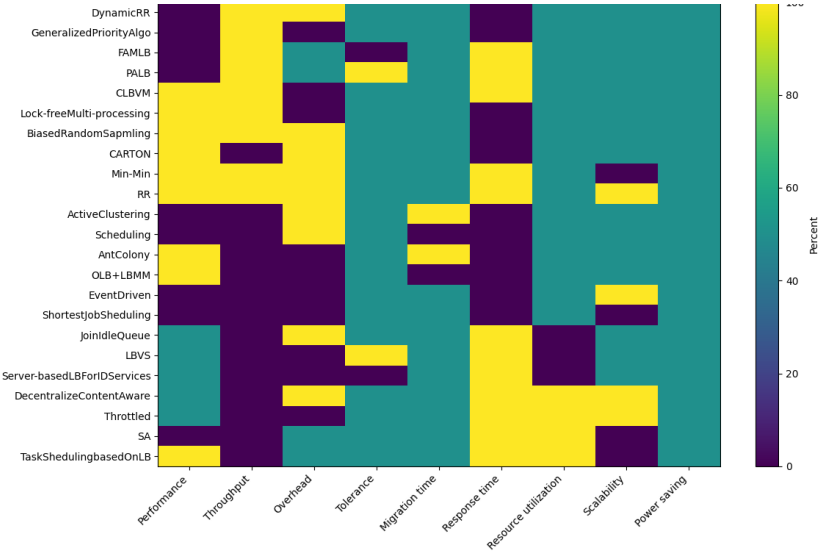

Fig. 6: Clustering of QoS criteria against selected algorithms.

indicates a focus on execution efficiency rather than fault handling.

Moreover, this study is particularly valuable because it explains why the model selects an algorithm: PALB because of energy efficiency, DynamicRR because of responsiveness, and FAMLB for migration-aware scheduling. Throttled due to its scalability and utilization optimization, RoundRobin, including performance-oriented simple balancing, finally joins IdleQueue for overhead reduction.

Combining AI tools with load balancing can significantly enhance cloud performance. Developing an accurate AI-driven LLM requires building expert models that focus on specific targets rather than relying solely on general purposes. The results demonstrate flexibility; however, they may vary under certain conditions. Therefore, expert models represent the most suitable approach. These models assist users in cloud computing, in our case, by selecting the most appropriate load-balancing technique according to their requirements. Furthermore, combining multiple models

can improve the optimization process.

Experts usually assist users in addressing their needs in a specific domain. Since the data were collected primarily from engineers and researchers in the domain, training multiple models offered a solution by enabling the selection of the most appropriate model to meet users' needs. The experiment encouraged a new perspective on investigating multiple models and combining various tools, which can lead to improved decision-making and useful assessments that help users respond to their needs in a specific domain. In addition, this evaluation emphasizes the ability to rely on and combine several tools at the same time, even manually or automatically via an AI agent, for instance.

6. CONCLUSION AND FUTURE WORK

Selecting an appropriate load balancing strategy remains a challenging task in cloud computing due to the diversity of Quality of Service (QoS) requirements and the dynamic nature of cloud environments. This paper presents an evaluation of several models, alongside our customized models, to select the appropriate approach for load balancing in cloud computing. The study emphasizes the necessity of using expert models specifically designed for the field, rather than relying on general models like ChatGPT and Copilot. Our findings suggest that tailored solutions hold significant value, as they align more closely with the specific requirements of cloud environments. As future work, we plan to extend the dataset by incorporating additional static, dynamic, and AI-based load balancing algorithms together with a broader range of cloud workloads and QoS metrics. Also intending to compare the study to other machine learning methods.

7. REFERENCES

[1] Wayne Xin Zhao et al. A survey of large language models. *Frontiers of Computer Science*, 17(6):670651, 2023.

[2] N. Yang, M. Fan, W. Wang, and H. Zhang. Decision-making large language model for wireless communication: a comprehensive survey on key techniques. *IEEE Communications Surveys & Tutorials*, vol. 28, pp. 3055–3088, 2026. doi: 10.1109/COMST.2025.3614494.

[3] Peter Mell and Timothy Grance. The nist definition of cloud computing. *NIST Special Publication 800-145*, 2011.

[4] Y. Pei et al. Llm-based cost-aware task scheduling for cloud computing systems. *Journal of Cloud Computing*, 14:1–20, 2025.

[5] M. S. M. Pozi and Y. Sato. A data-augmented model routing framework for efficient LLM deployment in edge–cloud environments. *The Journal of Supercomputing*, vol. 81, Article 1573, 2025. doi: 10.1007/s11227-025-08034-8.

[6] Anouar Ben Halima and Hafssa Benaboud. Challenges of load balancing algorithms in cloud computing utilizing data mining tools. *International Journal of Electrical and Computer Engineering (IJECE)*, 15(3):3449–3457, June 2025.

[7] R. M. Singh, L. K. Awasthi, and G. Sikka. Towards metaheuristic scheduling techniques in cloud and fog: an extensive taxonomic review. *ACM Computing Surveys*, vol. 55, no. 3, Article 50, pp. 1–43, 2022. doi: 10.1145/3494520.

[8] Anouar Ben Halima and Hafssa Benaboud. On combining clustering and classification algorithms to enhance load balancing in cloud computing. In *Digital Technologies and Applications*, volume 1640 of *Lecture Notes in Networks and Systems*, pages 3–13. Springer, 2026.

[9] X. Yu et al. Multi-objective routing for cloud-edge large language model services. *arXiv preprint arXiv:2507.15553*, 2025.

[10] J. Li, C. Xu, L. Jia, F. Wang, C. Zhang, and J. Liu. EACO-RAG: Towards distributed tiered LLM deployment using edge-assisted and collaborative RAG with adaptive knowledge update. *arXiv preprint arXiv:2410.20299*, 2024. doi: 10.48550/arXiv.2410.20299.

[11] A. Tekawade and S. Banerjee. A cost effective reliability aware scheduler for task graphs in multi-cloud system. *arXiv preprint arXiv:2212.09166*, 2022. doi: 10.48550/arXiv.2212.09166.

[12] N. Pasha, A. Agarwal, and R. Rastogi. Round robin approach for vm load balancing algorithm in cloud computing environment. *International Journal of Advanced Research in Computer Science and Software Engineering*, 4(5):34–39, 2014.

[13] R. Mishra. Ant colony optimization: a solution of load balancing in cloud. *International Journal of Web & Semantic Technology*, 3(2):33–50, 2012.

[14] S. Sethi. Efficient load balancing in cloud computing using fuzzy logic. *IOSR Journal of Engineering*, 2(7):65–71, 2012.

[15] M. S. Q. Z. Nine, M. A. K. Azad, S. Abdullah, and M. R. Rahman. Fuzzy logic based dynamic load balancing in virtualized data centers. In *2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–7, 2013. doi: 10.1109/FUZZ-IEEE.2013.6622384.

[16] G. Michael, K. L. Smith, and S. S. Vrbsky. Power-aware load balancing for cloud computing. In *Lecture Notes in Engineering and Computer Science*, volume 2193, pages 127–132, 2011.

[17] V. Nae, R. Prodan, and T. Fahringer. Cost-efficient hosting and load balancing of massively multiplayer online games. In *Proceedings of the IEEE/ACM International Workshop on Grid Computing*, pp. 9–16, 2010. doi: 10.1109/GRID.2010.5697956.

[18] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. R. Larus, and A. Greenberg. Join-idle-queue: A novel load balancing algorithm for dynamically scalable web services. *Performance Evaluation*, 68(11):1056–1071, 2011.

[19] R. K. Mondal, E. Nandi, and D. Sarddar. Load balancing scheduling with shortest load first. *International Journal of Grid and Distributed Computing*, 8(4):171–178, 2015.

[20] D. A. Agarwal and S. Jain. Efficient optimal algorithm of task scheduling in cloud computing environment. *International Journal of Computer Trends and Technology*, 9(7):344–349, 2014.

[21] Z. C. Chaczko, V. Mahadevan, S. Aslanzadeh, and C. McDermid. Availability and load balancing in cloud computing. In *International Conference on Computer and Software Modeling (ICCSM 2011)*, International Proceedings of Computer Science and Information Technology (IPCSIT), vol. 14, pp. 134–140, 2011.

[22] M. Randles, D. Lamb, and A. Taleb-Bendiab. Experiments with honeybee foraging inspired load balancing. In *Proceedings of the International Conference on*

Developments in eSystems Engineering (DeSE 2009), pp. 240–247, 2009. doi: 10.1109/DeSE.2009.19.

[23] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal, and S. Dam. A genetic algorithm (ga) based load balancing strategy for cloud computing. *Procedia Technology*, 10:340–347, 2013.

[24] B. Mondal, K. Dasgupta, and P. Dutta. Load balancing in cloud computing using stochastic hill climbing: a soft computing approach. *Procedia Technology*, 4:783–789, 2012.

[25] Y. Fang, F. Wang, and J. Ge. A task scheduling algorithm based on load balancing in cloud computing. In *Lecture Notes in Computer Science*, volume 6318, pages 271–277, 2010.

[26] N. J. Kansal and I. Chana. Cloud load balancing techniques: a step towards green computing. *International Journal of Computer Science Issues*, 9(1):238–246, 2012.

[27] S. Sidhu and M. Mittal. A new era to balance the load on cloud using vector dot load balancing method. *International Journal of Technology and Computing*, 2(4):413–418, 2016.

[28] V. K. Reddy, K. Deva Surya, M. Sai Praveen, B. Lokesh, A. Vishal, and K. Akhil. Performance analysis of load balancing algorithms in cloud computing environment. *Indian Journal of Science and Technology*, 9(18), 2016.

[29] N. Xuan Phi, C. T. Tin, L. N. Ky Thu, and T. C. Hung. Proposed load balancing algorithm to reduce response time and processing time on cloud computing. *International Journal of Computer Networks & Communications*, 10(3):87–98, 2018.

[30] H. Mehta, P. Kanungo, and M. Chandwani. Decentralized content aware load balancing algorithm for distributed computing environments. In *Proceedings of the International Conference & Workshop on Emerging Trends in Technology (ICWET '11)*, pp. 370–375, 2011. doi: 10.1145/1980022.1980102.

[31] A. Bhadani and S. Chaudhary. Performance evaluation of web servers using central load balancing policy over virtual machines on cloud. In *COMPUTE 2010 - The 3rd Annual ACM Bangalore Conference*, pages 1–4, 2010.

[32] X. Liu, L. Pan, C.-J. Wang, and J.-Y. Xie. A lock-free solution for load balancing in multi-core environment. In *2011 3rd International Workshop on Intelligent Systems and Applications (ISA)*, pp. 1–4, 2011. doi: 10.1109/ISA.2011.5873313.

[33] T. Kokilavani and D. I. George Amalarethinam. Load balanced minmin algorithm for static meta-task scheduling in grid computing. *International Journal of Computer Applications*, 20(2):42–48, 2011.