Scalable and Optimized Load Balancing in Cloud Systems: Intelligent Nature-Inspired Evolutionary Approach

Akhil Reddy Duggasani[1]

[1]Independent Researcher San Jose, USA

Publication Date: 2025/05/28

Abstract: The optimal system performance depends on efficient scheduling of numerous virtualized resources which Cloud computing orchestrates. Organizations using cloud computing require efficient task scheduling to achieve optimal system performance because the platform includes multiple virtualized resources. This paper proposes a novel Hybrid Lyrebird Falcon Optimization Algorithm (HLFOA) for global exploration and the Falcon Optimization Algorithm (FOA) for local exploitation. Through HLFOA virtual machine (VM) tasks become better distributed across sites while achieving minimum makespan together with reduced power usage and enhanced CPU resource utilization. Performance analysis with CloudSim 4.0 simulation proves that HLFOA is more efficient than baseline methods as PSO. At 100 tasks, HLFOA achieves a makespan of 299 units, compared to PSO's 513 units, and at 500 tasks, it reduces makespan to 2015 units, while PSO reaches 3868 units. The adoption of HLFOA improves both system energy consumption efficiency and processor utilization levels. HLFOA shows promise as a scalable and effective solution for cloud load balancing, which enables robust optimization of cloud resource allocation.

Keywords: *Cloud computing, Load balancing, Virtual Machines (VMs), Virtual Machines (VMs), Nature-Inspired, Resource Allocation, Hybrid Lyrebird Falcon Optimization Algorithm.*

How to Cite: Akhil Reddy Duggasani (2025) Scalable and Optimized Load Balancing in Cloud Systems: Intelligent Nature-Inspired Evolutionary Approach *International Journal of Innovative Science and Research Technology*, 10(5), 2153-2160. https://doi.org/10.38124/ijisrt/25may1290

I. INTRODUCTION

Modern industries have experienced major transformations in their data processing while storage and retrieval methods due to rapid technological advancement. The technological revolution created new computing models where cloud computing stands as the fundamental infrastructure of current-day IT systems [1]. Digital service delivery and consumption methods transformed with cloud computing since users can access shared internet-based computing resources for scalable, flexible, and cost-efficient operations [2]. The increasing usage of cloud platforms requires efficient resource management solutions because these solutions ensure system stability and peak performance [3]. The increased demand for cloud computing has expanded the importance of optimization algorithms that solve problems about performance optimization and resource management and cost reduction [4][5]. Cloud performance relies heavily on load balancing since it determines both speed and efficiency of responses to user input [6][7].

The term "load balancing" is used to describe how workloads are distributed fairly among the available VMs in data centers in order to avoid resource overload and underutilization [8][9][10]. Quality of Service (QoS) remains consistent and system downtime decreases while user experience improves when load distribution reaches its best level. Nevertheless, the traditional load balancing techniques are inadequate to the dynamic and elastic cloud infrastructure and are not scalable [11][12]. The absence of adequate solutions for real-time operations under work quantity alterations necessitates more adaptive intelligent solutions [3].

Researchers now focus on evolutionary algorithms based on nature patterns because these optimization strategies model biological processes like evolution and swarm activity and foraging operations [13][14]. There are certain merits possessed by the above said algorithms like parallelism, flexibility, and robustness that qualifies them for solving the dynamic optimization problems in cloud environment [15][16]. It can flexibly decide the execution times of tasks and distribute jobs, with an ability to regulate Load Balancing under different situations. Further to this, the application of nature-inspired evolutionary strategies on the cloud load balancing system provides a way forward on how resource utilization, scalability and system efficiency can be achieved.

➢ *Motivation and Contributions*

The research bases its work on the problems found in cloud computing that include poor load balancing alongside high energy consumption and underused resources. Dynamic workloads cause most existing algorithms to fail resulting in energy expenditure delays.

- The research presents the HLFOA solution as a method for improving task scheduling by enhancing makespan performance alongside energy efficiency and CPU utilization. This paper has various contributions that listed below

- The researcher presents a new optimization technique named Hybrid Lyrebird Falcon Optimization Algorithm (HLFOA which unites LOA and FOA for enhancing cloud computing task scheduling efficiency.

- An effective divergent method creation is designed to equally share workload tasks between virtual machines which avoids computational delays.

- A probability-based switching mechanism has been developed to manage the exploration of global possibilities and local optimization while the optimization process occurs.

- The researcher develops mathematical models for measuring performance indicators that include makespan together with Energy Consumption and CPU utilization.

➢ *Justification and Novelty*

The research basis stems from the necessity to develop improved load balancing methodologies for cloud environments due to inadequate performance of conventional techniques. The novelty of the proposed work is the development of the Hybrid Lyrebird Falcon Optimization Algorithm (HLFOA), which uniquely combines the global search capabilities of LOA with the local refinement strength of FOA. This hybrid approach, guided by a dynamic switching mechanism, enhances both exploration and exploitation, resulting in improved task scheduling, reduced makespan, lower energy consumption, and better CPU utilization compared to existing methods.

➢ *Structure of Paper*

Here is how the paper is organized: Section II discusses optimization strategies for cloud load balancing that draw inspiration from nature. The suggested framework based on HLFO is described in Section III. The examination of comparisons is presented in Section IV. Section V delves into the study and how it may shape future investigations.

II. LITERATURE REVIEW

This section reviews optimizing Load Balancing within Cloud Computing, the effectiveness of a nature-inspired evolutionary approach. The summary of related work is based on parameters, features are provided in Table I,

Priya and Perumal (2024) Cloud computing is expanding faster than anticipated; hence, load balancing must be done effectively. Weighted Least Connection (WLC). It comprises adaptive weight adjustments, load checking in real time, and task allocation based on server capacity to improve resilience and scalability. The WLC system reduces latency up to 25% and improves resource utilization across the servers by 41.67%. Medium and large task times also show an improvement of 33.33%-time savings for completion of a task. These highlights the ability of WLC in balancing cloud environments, and dynamic load balancing techniques as an alternative to static load balancing methods [17].

Suresh et al. (2024) provide a fault-tolerant load balancing strategy that outperforms state-of-the-art robust optimisation approaches by optimising it using MCSOFLB, a multi-objective cat swarm algorithm. The results of the experiments prove beyond a reasonable doubt that the method you proposed is superior. When compared to existing benchmark algorithms, the MCSOFLB approach improves upon them in the following ways: average throughput (32%), makespan (31%), cost (12%), success rate (6%), and resource utilisation (6%) [18].

Khaleel (2024) running simulations with the help of the CloudSim toolbox, the outcomes highlight the RASA algorithm's dominance with regard to its convergence speed, task placement precision, and workload distribution fairly. Reduced Latency overhead by 9%, Processing Time by 14%, workload imbalance by 15%, Energy Consumption by 19%, and idle times by 26% are all results of their suggested approach. In addition, it improves efficiency by 27% and resource availability by 22%, all while increasing service throughput by 32% [19].

Javadpour et al. (2023) recommended a method for sorting jobs according to when they are due for completion. They also sort the actual machines according to their current configuration. The suggested approach now moves the tasks to nearby physical machines that have the same priority class. Also, by employing the DVFS approach, they lower the energy consumption of the machines that handle the low-priority jobs. To keep the workload balanced, the suggested technique moves the tasks. It also determines if the machines' class changed based on their results. Using the CloudSim package, they have assessed and verified the suggested approach. The outcomes of a simulation show that the suggested approach reduced power usage by 20% and energy consumption by 12% [20].

Kamila et al. (2022) suggest and incorporate cloud-based AI machine learning strategies with the idea of high-performance computing. The proposed integrated design approach has been evaluated on a range of tasks and decisions grounded on ML classification and regression models with the aim of automatically enhancing the system's performance during real run-time instances. The simulated results of their ML integrated design demonstrate that, in comparison to current non-ML-based design models, it is 38.15 percent quicker in terms of failure point recovery and saves 7.5 percent in business costs [21].

Ajagbe et al. (2022) design innovative cloud-based load balancing solutions. Topics such as reaction time and throughput were tested in the trials. Honeybee, PSO, SASOS, round-robin, PSO-ACO, and P-ACOHONEYBEE all had reaction times of 2791, 2780, 2784, 2767, 2727, and 2599 milliseconds, commensurately. The results of the throughput tests for honeybee, PSO, SASOS, round-robin, PSO-ACO, and P-ACOHONEYBEE are 7451, 7425, 7398, 7357, 7387, and 7482 bps, respectively. Among the 10 nodes, the P-ACOHONEYBEE method yields the best outcomes in terms of reaction time, throughput, and general performance [22].

Shafiq et al. (2021) The Service Level Agreement (SLA) is a contract that cloud developers provide to consumers, and it lays out the criteria for activities like scheduling and load balancing. The LB algorithm takes into account crucial SLA criteria like Deadline. The suggested method for optimizing resources and enhancing load balancing while taking into consideration the priorities of VMs), Resource Allocation, and QoS job factors. In contrast to the current Dynamic LBA algorithm, the suggested LB algorithm yields an average resource utilisation of 78%, according to the data. With reduced Execution time and Makespan, it also achieves excellent performance [23].

Various load balancing algorithms for cloud computing that include WLC, MCSOFLB, and RASA have successfully enhanced several operational aspects including resource utilization, fault tolerance, energy efficiency and task priority management. Existing load balancers in cloud computing several drawbacks including underperformance with changing workloads alongside difficulty in achieving SLA and insufficient real-time modification capabilities. Most available approaches fail to utilize machine learning techniques for dynamic task scheduling in their entirety.

Table 1 Comparative Analysis of Load Balancing Approaches in Cloud Computing using Nature-Inspired and Intelligent Techniques

Author(s)	**Objectives**	**Methodology**	**Parameters**	**Features**
Priya and Perumal (2024)	Weighted Least Connection (WLC)	Adaptive weight adjustment, real-time load monitoring, server-capacity-based allocation	Reduced latency by 25%; 41.67% improvement in resource utilization; 33.33% task time savings	Demonstrated dynamic balancing benefits over static methods; lacks testing on diverse workload types
Suresh et al. (2024)	MCSOFLB (Multi-objective CatSwarm Optimization for Fault-Tolerant Load Balancing)	Fault tolerance, multi-objective optimization	31% makespan, 6% resource use, 12% cost, 6% success rate, 32% throughput improvement	Outperforms benchmarks; complex implementation may hinder real-time adoption
Khaleel (2024)	RASA (Resource-Aware Scheduling Algorithm)	Task placement precision, fair workload distribution	Reduces latency by 9%, processing time by 14%, energy by 19%, idle time by 26%; increases efficiency by 27%, throughput by 32%	Efficient and eco-friendly; real-world validation still required
Javadpour et al. (2023)	Deadline-Aware Scheduling with DVFS	Task prioritization by deadline, energy efficiency via DVFS, task migration	12% reduction in energy use, 20% reduction in power use	Energy optimization emphasized; focus on task migration under physical constraints
Kamila et al. (2022)	ML-based High-Performance Load Balancing	Integration of ML for regression/classification, self-correcting performance at runtime	38.15% faster failure recovery, 7.5% cost savings	Proactive and adaptive; implementation dependent on accurate ML models
Ajagbe et al. (2022)	P-ACOHONEYBEE (Hybrid PSO-ACO-Honeybee)	Combines three nature-inspired algorithms with mathematical optimization	Lowest response time (2599ms), highest throughput (7482 bps) among tested methods	Efficient under test; broader scalability not explored
Shafiq et al. (2021)	SLA-Aware Load Balancing	QoS optimization, VM prioritization, resource allocation based on SLA parameters	78% resource utilization, reduced execution time and makespan	Focus on SLA compliance; limited exploration of multi-objective load contexts

III. METHODS AND MATERIALS

The methodology of this study involves designing and implementing an efficient Load Balancing framework in a Cloud Computing environment employing a novel HLFOA. The proposed Hybrid Lyrebird Falcon Optimization Algorithm (HLFOA) is initialized by generating a population of candidate solutions within defined lower and upper bounds. Next, the algorithm dynamically switches between the Lyrebird Optimization Algorithm (LOA) for global exploration utilizing hiding and fleeing behaviors and the Falcon Optimization Algorithm (FOA) for local exploitation, leveraging sine-based and cosine-based search strategies. Afterwards, important performance indicators, including makespan, energy usage, and CPU utilisation, are assessed, and the jobs are allocated to virtual machines according to HLFOA's optimised scheduling recommendations. Lastly, it put the findings up against other methods that have been developed to enhance the effectiveness of load balancing in cloud computing. The workflow of implementation of cloud environment are display in Figure 1.

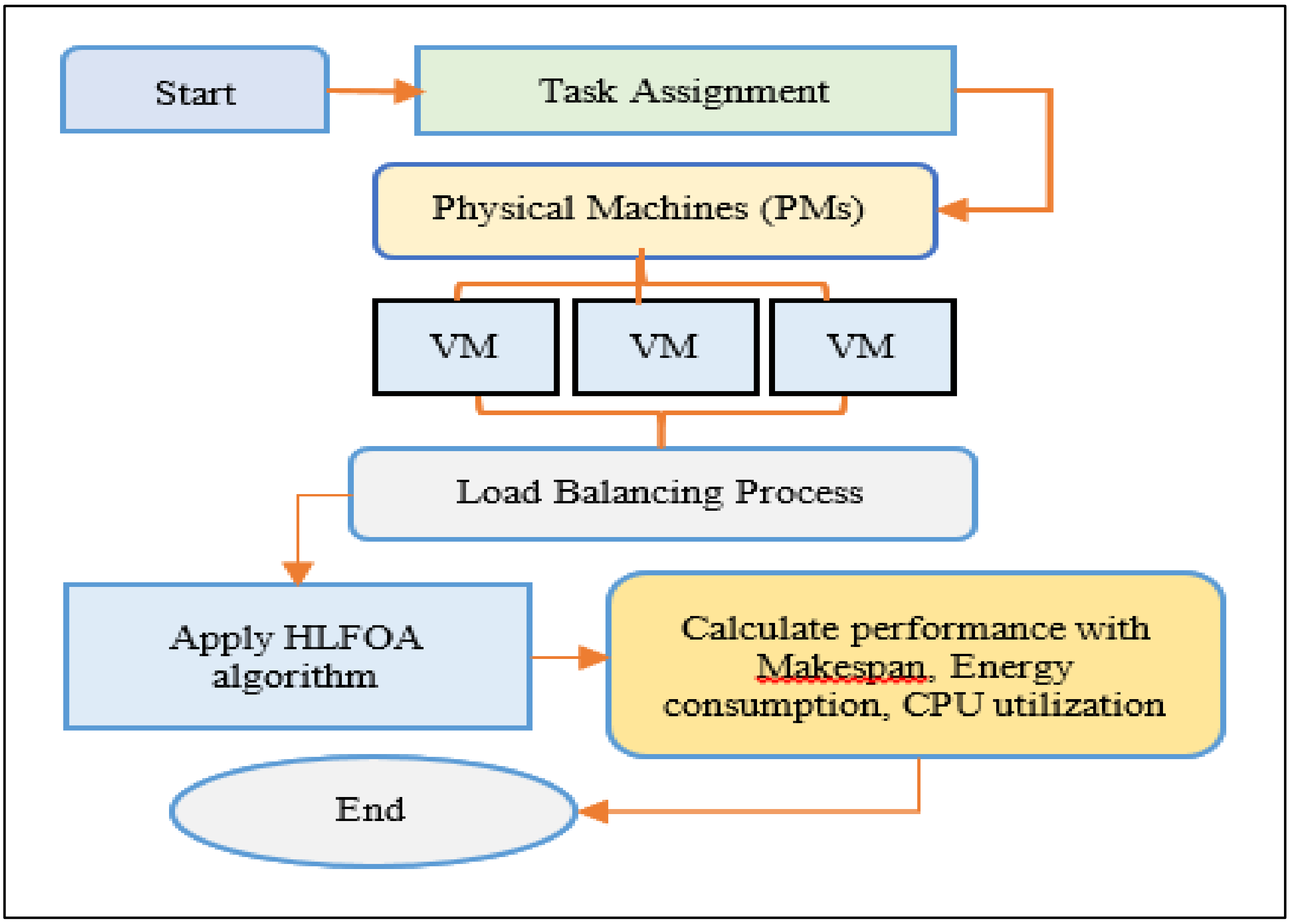

Fig 1 Flowchart for Load balancing in Cloud Computing

Each process and steps of proposed flowchart are explained below:

➢ *Virtual Machine*

The suggested effort is focused on optimizing load calculation and VM clustering in a cloud environment via Load Balancing across Physical Machines (PMs) and Virtual Machines (VMs).

➢ *Proposed Hybrid Lyrebird Falcon Optimization Algorithm*

The HLFOA combines the strengths of the LOA and the FOA. This hybridisation improves the balance between exploration and exploitation in the search space. HLFOA starts by initializing a population of candidate solutions within the defined lower and upper bounds. It gives as Equation (1):

$$x_i = lb + rand.(ub - lb) \qquad (1)$$

Here, x_i is the position of the ith candidate solution, lb and ub represent the lower and upper bounds of the variables, and rand is a number between zero and one that is distributed equally. In the Lyrebird phase, two types of behaviors are modeled: hiding and fleeing. The hiding behavior moves the candidate toward the best solution and a randomly selected peer, encouraging global exploration. It is formulated in Equation (2):

$$x_i = x_i + r_i.(x_{best} - x_i) + r_r.(x_{rand} - x_i) \qquad (2)$$

In this Equation (2), x_{best} is the best solution found so far, x_{rand} is a randomly selected solution, and $r_i, r_2 \in [0,1]$ are random values that control the influence of these directions. The fleeing behavior moves the candidate away from the worst-performing solution, which helps avoid poor regions of the search space and further promotes exploration.

$$x_i = x_i + r_3.(x_i - x_{worst}) + r_4.(x_{rand} - x_i) \qquad (3)$$

Here, x_{worst} represents the worst solution in the population, and $r_3, r_4 \in [0,1]$ are random coefficients for scaling the fleeing direction in Equation (3). On the other hand, the Falcon Optimization phase focuses on exploiting promising regions. The target pursuit behavior allows a candidate to move toward the best solution in a sine-shaped trajectory, which helps refine solutions around optima.

$$x_i = x_i + r_5.sin(\theta).(x_{best} - x_i) \qquad (4)$$

In this Equation, $\theta \in [0, 2\pi]$ is a random angle that introduces variation in the movement, and r_5 is a random coefficient for controlling step size illustrate in Equation (4). The diving behavior guides the candidate toward the average position of the population using a cosine-based trajectory. This supports local convergence by focusing the search near the centroid of the population. It gives as Equation (5)

$$x_i = x_i + r_6.Cos(\theta).(x_i - \bar{x}) \qquad (5)$$

Here, $\bar{x}$ is the mean of all candidate positions, and $r_6, \in [0,1]$ is a random scaling factor. To combine both LOA and FOA, HLFOA uses a probabilistic switch that determines which behavior to apply at each iteration. This selection is governed by a dynamic probability $p\ (t)$ which decreases over time to allow more exploration in the early stages and more exploitation in later stages. It gives as Equation (6):

$$x_i = \begin{cases} LOA\ behavior & if\ rand < p(t) \\ FOA\ behavior & otherwise \end{cases} \qquad (6)$$

The control parameter p(t) is defined as Equation (7):

$$p(t) = 1 - \frac{t}{T} \qquad (7)$$

where T is the most iterations possible and t is the number of iterations that are currently occurring. This ensures a smooth transition from exploration to exploitation. In summary, HLFOA benefits from the global search ability of LOA and the local refinement of FOA. It is highly adaptable and suitable for solving a wide range of optimization problems, including feature selection, classification, and real-time decision-making tasks.

➢ *Performance Metrics*

The following performance measures are calculated for Load Balancing in Cloud Computing. These parameters are formulated below:

- *Makespan*

Make span as the total execution time of all the tasks. The workflow makespan, which refers to the maximum finish time of all the task by considering both the task [24] execution time and the data transfer time among tasks. This optimization objective is formulated as follows Equation (8):

$$Minimize f_2(x) = \max_{v_i \in V} ft(v_i, *) \qquad (8)$$

- *Energy Consum*

The total time it takes for the tasks to complete is determined by determining the maximum time needed for the VM, which is running simultaneously.

- *ption*

The next crucial factor from the viewpoints of cloud providers and users is energy consumption [25]. There are two main components to energy usage in the cloud paradigm: computation energy and idle energy. It gives as Equation (9):

$$e_{con}(vm^k) = \int_0^k e_{con}^{com}\ (vm^k t) + e_{con}^{idle}\ (vm^k t) dt \qquad 9$$

The virtual machine is immediately terminated when it is not in use. The data center's policy determines whether or not the mathematical model for power idle is necessary. You may disable it for datacenters that do it when a virtual machine or host is no longer needed.

- *CPU Utilization (CU)*

CPU utilisation is a metric for evaluating an efficiency of CPU utilisation in Cloud Computing settings. To prevent certain PMs or VMs from being overworked while others sit idle, CPU utilisation balancing is implemented. Choosing the right CPU allows you to avoid performance bottlenecks and make the most of your hardware resources. A common formula used to calculate CPU utilization is Equation (10):

$$CU = \left(1 - \frac{CPU\ Idle\ time}{Total\ time}\right) \times 100\% \qquad 10$$

Where CPU idle time is Time when the CPU is not executing any process. And the total time is the Total observation period (includes idle and active time).

IV. RESULT ANALYSIS AND DISCUSSION

This section details the results of the cloud system's experimental evaluation of the suggested load-balancing method. The experimental setup involves a simulated cloud environment using CloudSim 4.0 as the simulation toolkit and Python 3.11.1 for implementation. The simulation is conducted with a single physical machine (PM) and a varying number of VMs, ranging from 10 to 50. The experiments use between 100 and 500 processed tasks to study system performance under varying workload conditions. The evaluation of the proposed algorithm considered makespan and energy consumption and CPU utilization as indicators to measure load balancing efficiency effectiveness. The experiment used different task quantities to determine workload intensity effects on these performance measures as displayed in Table II.

Table 2 Experiment Result of Proposed HLFO Algorithm for Load Balancing in Cloud Computing

Task count	**Makespan**	**Energy consumption**	**CPU Utilization**
100	299	42.724	0.003376
200	1013	45.171	0.004606
300	1546	48.273	0.0012462
400	1972	55.412	0.012035
500	2015	61.978	0.00934

A Cloud Computing environment demonstrates the connection between task count and Energy Consumption as shown in Figure 2. As the task count increases from 100 to 500, energy consumption rises steadily from 42.724 to 61.978 units. Energy consumption rises in a linear fashion according to the data which shows that increased task numbers generate additional computational workloads that enhance power consumption demands thus stressing the necessity of energy-efficient scheduling in cloud systems.

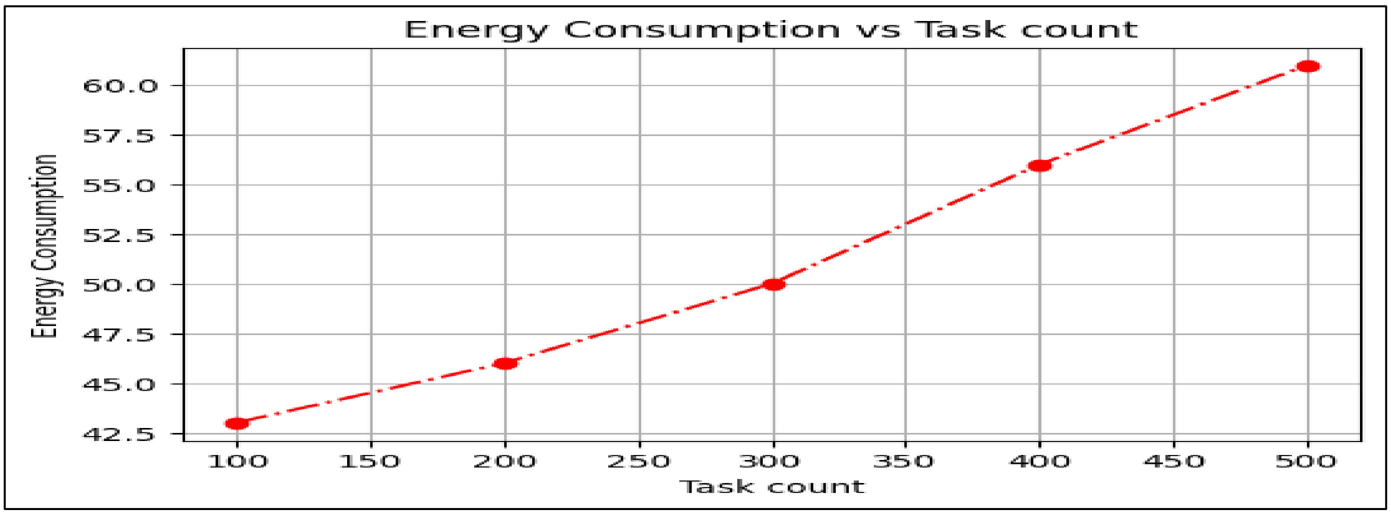

Fig 2 Plot for Energy Consumption

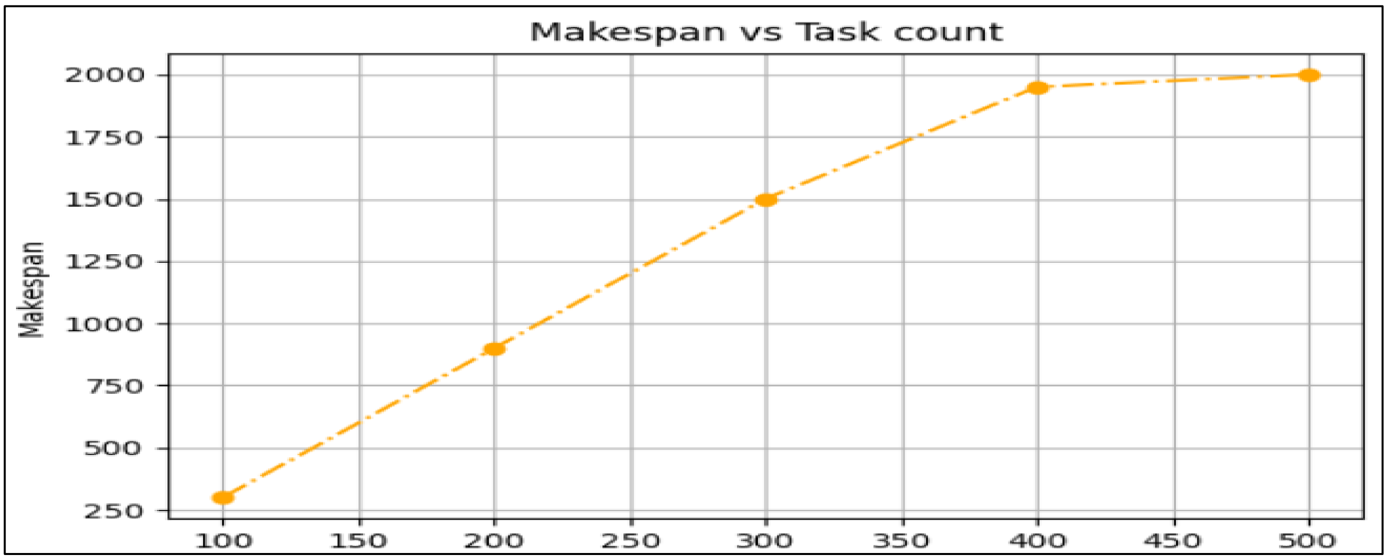

Fig 3 Plot for Makespan

Figure 3 depicts the variation of makespan with respect to task count. The data reveals a dramatic rise in makespan duration from 299 until 2015 units as the number of tasks extends from 100 up to 500 showing a sustained non-linear time increase in total processing duration. The trend shows that larger task batches cause higher computational delays, stressing the need for efficient scheduling to reduce makespan and enhance responsiveness in cloud systems.

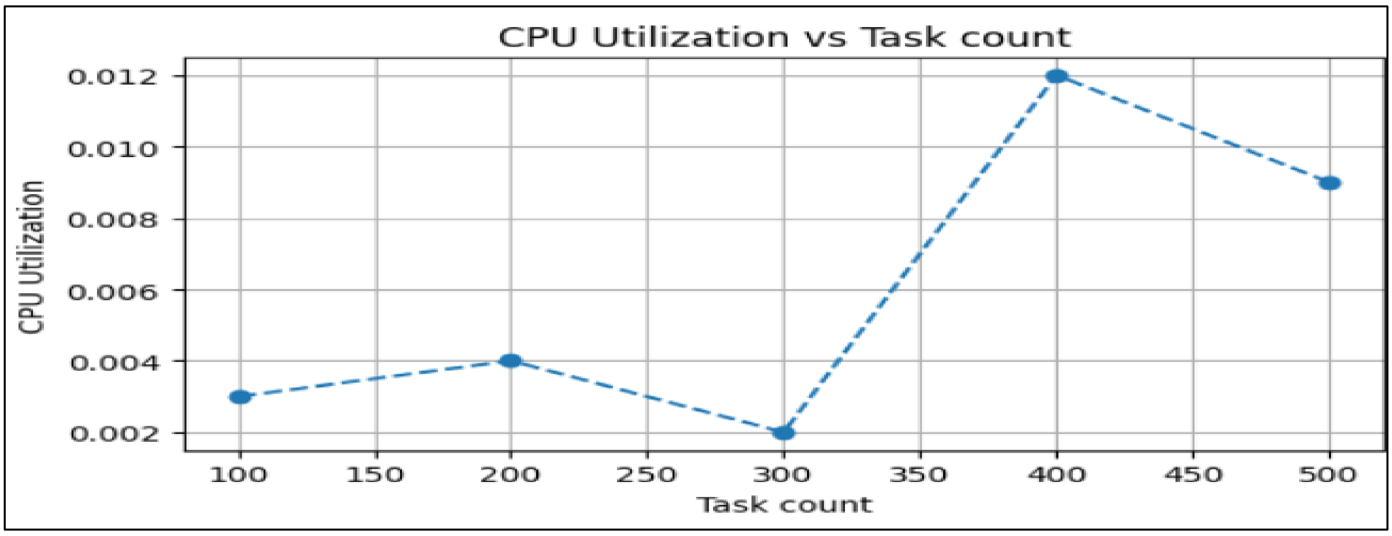

Fig 4 Task Count Vs CPU Utilization

Figure 4 illustrates the relationship between task count and CPU utilization, serving as a proxy for load balancing efficiency in a cloud computing environment. Unlike the linear trends observed in makespan and energy consumption, CPU utilization exhibits a non-linear and fluctuating pattern as task count increases from 100 to 500. The utilization initially rises, dips significantly at a task count of 300, peaks at 0.012035 for a task count of 400, and then slightly decreases. Despite this variability experts believe it stem from unusual distribution patterns of workloads which emphasizes the necessity of dynamic load balancing systems for sustaining stable performance under different workload conditions.

➢ *Comparison and Discussion*

The suggested method is contrasted with current scheduling and load-balancing approaches in the comparative analysis section. The following comparison of algorithms are based on load balancing parameters are illustrated in Table III.

The experimental results show HLFO delivers lower makespan values than PSO while performing at all identified task levels. For instance, at 100 tasks, HLFO records a makespan of 299 compared to PSO’s 513, and at 500 tasks, HLFO achieves 2015 versus PSO’s 3868. The HLFO algorithm demonstrates reliable performance at improving operational efficiencies along with minimizing computational delays throughout cloud system environments.

Table 2 Makespan Comparison Between Propose and Base Algorithm Across Task Count

Task count	HLFO	PSO[26]
100	299	513
200	1013	1620
300	1546	2522
400	1972	3374
500	2015	3868

Hybrid Lyrebird Falcon Optimization Algorithm (HLFOA bring multiple benefits which include enhanced load distribution alongside shorter execution time along with minimized power usage. HLFOA achieves workload adaptation and virtual machine task distribution efficiency through its implementation of LOA global exploration and FOA local exploitation. The system uses minimal energy while maximizing CPU capacity to solve essential operational problems which affect cloud computing environments. The exploration to exploitation dynamic switch in PSO provides a scalable solution that performs beyond traditional PSO algorithms in resource optimization for task scheduling.

V. CONCLUSION AND FUTURE SCOPE

A major obstacle in the Cloud Computing environment is effective task scheduling, which is becoming more of a problem due to rising task quantities and unpredictable resource needs. This paper introduces the HLFOA for efficient Load Balancing in Cloud Computing environments. The HLFOA applies LOA's worldwide exploration abilities together with FOA's local optimization tasks to improve scheduling results. The simulated findings demonstrate that HLFOA achieves superior performance in key performance outcomes compared to conventional algorithms PSO. Specifically, HLFOA achieves a lower makespan, with task execution time reduced from 513 units (using PSO) to 299 units at 100 tasks, and from 3868 units to 2015 units at 500 tasks. The HLFOA approach demonstrates superior energy savings together with better CPU resource utilization which confirms its capacity to optimize cloud environment efficiency. Fixing scheduling problems in volatile cloud systems remains a challenge for HLFOA even though it delivers substantial performance gains in resource management. HLFOA requires additional optimization for its large-scale system scalability.

The future work should be aimed at improving HLFOA for working in multi-Cloud and edge computing scenarios as well as the introduction of machine learning to provide scheduling predictions on task loads and implement them to make the algorithm more dynamic in managing Cloud resources.

REFERENCES

[1]. S. Murri, “Data Security Challenges and Solutions in Big Data Cloud Environments,” Int. J. Curr. Eng. Technol., vol. 12, no. 6, 2022, doi: https://doi.org/10.14741/ijcet/v.12.6.11.

[2]. H. Talabani, “Machine learning-based cloud computing and IOT: A review,” Int. J. Cloud Comput. Database Manag., vol. 5, 2024, doi: 10.33545/27075907.2024.v5.i2b.72.

[3]. S. Murri, S. Chinta, S. Jain, and T. Adimulam, “Advancing Cloud Data Architectures: A Deep Dive into Scalability, Security, and Intelligent Data Management for Next-Generation Applications,” Well Test. J., vol. 33, no. 2, pp. 619–644, 2024, [Online]. Available: https://welltestingjournal.com/index.php/WT/article/view/128

[4]. S. S, R. R. Patil, and P. C, “Cloud computing an overview,” Int. J. Eng. Technol., vol. 7, no. 4, p. 2743, Oct. 2018, doi: 10.14419/ijet.v7i4.10904.

[5]. S. Arora and S. R. Thota, “Automated Data Quality Assessment And Enhancement For Saas Based Data Applications,” J. Emerg. Technol. Innov. Res., vol. 11, pp. i207–i218, 2024, doi: 10.6084/m9.jetir.JETIR2406822.

[6]. L. Golightly, V. Chang, Q. A. Xu, X. Gao, and B. S. C. Liu, “Adoption of cloud computing as innovation in the organization,” Int. J. Eng. Bus. Manag., 2022, doi: 10.1177/18479790221093992.

[7]. R. Kumar Verma, S. Singh, and Y. Mohan, “Importance Of Big Data And Cloud Computing

Techniques In Modern Scenario," J. Algebr. Stat., 2022.

[8]. S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: A big picture," Journal of King Saud University - Computer and Information Sciences. 2020. doi: 10.1016/j.jksuci.2018.01.003.

[9]. N. A. Joshi, "Technique for Balanced Load Balancing in Cloud Computing Environment," Int. J. Adv. Comput. Sci. Appl., 2022, doi: 10.14569/IJACSA.2022.0130316.

[10]. A. Kushwaha, P. Pathak, and S. Gupta, "Review of optimize load balancing algorithms in cloud," Int. J. Distrib. Cloud Comput., vol. 4, no. 2, pp. 1–9, 2016.

[11]. J. Zhou et al., "Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing," J. Cloud Comput., 2023, doi: 10.1186/s13677-023-00453-3.

[12]. N. Patel, Architecting the Future Mastering EVPN, VXLAN, SASE, and SD-Wan: In Cloud-Driven, Data Centers, Kindle. 2024.

[13]. A. S. Manekar and P. Gera, "Optimize Task Scheduling and Resource Allocation Using Nature Inspired Algorithms in Cloud based BDA," Webology, 2021, doi: 10.14704/WEB/V18SI01/WEB18049.

[14]. D. C. R. Aniket Deshpande Dr. Arshey M, Ravuri Daniel, Deepak Dasaratha Rao, Dr. Emerson Raja, "Optimizing Routing in Nature-Inspired Algorithms to Improve Performance of Mobile Ad-Hoc Network," Int. J. Intell. Syst. Appl. Eng., pp. 508–516, 2023.

[15]. N. Donyagard Vahed, M. Ghobaei-Arani, and A. Souri, "Multiobjective virtual machine placement mechanisms using nature-inspired metaheuristic algorithms in cloud environments: A comprehensive review," Int. J. Commun. Syst., 2019, doi: 10.1002/dac.4068.

[16]. A. Gogineni, "Novel Scheduling Algorithms For Efficient Deployment Of Mapreduce Applications In Heterogeneous Computing," Int. Res. J. Eng. Technol., vol. 4, no. 11, p. 6, 2017.

[17]. D. S. Priya and S. Perumal, "An Integrated Algorithm for Load Balancing Discrepancies in Cloud Computing Using Weighted Least Connection," in 2024 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), 2024, pp. 1–7. doi: 10.1109/ICSES63760.2024.10910374.

[18]. P. Suresh et al., "Optimized task scheduling approach with fault tolerant load balancing using multi-objective cat swarm optimization for multi-cloud environment," Appl. Soft Comput., vol. 165, p. 112129, Nov. 2024, doi: 10.1016/j.asoc.2024.112129.

[19]. M. I. Khaleel, "Region-aware dynamic job scheduling and resource efficiency for load balancing based on adaptive chaotic sparrow search optimization and coalitional game in cloud computing environments," J. Netw. Comput. Appl., 2024, doi: 10.1016/j.jnca.2023.103788.

[20]. A. Javadpour et al., "An Energy-optimized Embedded load balancing using DVFS computing in Cloud Data centers," Comput. Commun., 2023, doi: 10.1016/j.comcom.2022.10.019.

[21]. N. K. Kamila et al., "Machine learning model design for high performance cloud computing & load balancing resiliency: An innovative approach," J. King Saud Univ. - Comput. Inf. Sci., 2022, doi: 10.1016/j.jksuci.2022.10.001.

[22]. S. A. Ajagbe, M. O. Oyediran, A. Nayyar, J. A. Awokola, and J. F. Al-Amri, "P-ACOHONEYBEE: A Novel Load Balancer for Cloud Computing Using Mathematical Approach," Comput. Mater. Contin., 2022, doi: 10.32604/cmc.2022.028331.

[23]. D. A. Shafiq, N. Z. Jhanjhi, A. Abdullah, and M. A. Alzain, "A Load Balancing Algorithm for the Data Centres to Optimize Cloud Computing Applications," IEEE Access, 2021, doi: 10.1109/ACCESS.2021.3065308.

[24]. F. K. Karim, S. Ghorashi, S. Alkhalaf, S. H. A. Hamza, A. Ben Ishak, and S. Abdel-Khalek, "Optimizing makespan and resource utilization in cloud computing environment via evolutionary scheduling approach," PLoS One, vol. 19, no. 11 November, pp. 1–23, 2024, doi: 10.1371/journal.pone.0311814.

[25]. F. Kusumaningayu, "An Optimization on Task Scheduling for Makespan, Energy Consumption, and Load Balancing in Cloud Computing Using Meta-Heuristic," Int. J. Adv. Trends Comput. Sci. Eng., 2020, doi: 10.30534/ijatcse/2020/207942020.

[26]. D. Yu, Z. Xu, and M. Mei, "Multi-objective Task Scheduling Optimization Based on Improved Bat Algorithm in Cloud Computing Environment," Int. J. Adv. Comput. Sci. Appl., 2023, doi: 10.14569/IJACSA.2023.01406117.