International Journal of Data Science and Artificial Intelligence (IJDSAI)
Volume 02, Issue 04, July - August (2024)

RESEARCH ARTICLE

DYNAMIC LOAD BALANCING IN CLOUD COMPUTING USING HYBRID KOOKABURRA-PELICAN OPTIMIZATION ALGORITHMS

G. Saranya [1, *], G. Belshia Jebamalar [2] and Chukka Santhaiah [3]

[1] Department of Computer Science and Engineering, S.A. Engineering College, Poonamallee, Thiruverkadu, Tamil Nadu 600077 India.
[2] Department of Computer Science and Engineering, SA Engineering College, Poonamallee, Thiruverkadu, Tamil Nadu 600077 India.
[3] Department of Computer Science and Engineering, Sri Venkateswara College of Engineering, Tripati, Tamil Nadu India.

[*]*Corresponding e-mail: saran03ganesan@gmail.com*

Abstract – Cloud Computing (CC) technology facilitates virtualized computer resources to users via service providers. Load balancing assumes a critical role in distributing dynamic workloads across cloud systems, ensuring equitable resource allocation without overwhelming or underutilizing virtual machines (VMs). However, uneven workload distribution poses a significant challenge in cloud data centers, hindering efficient resource utilization. To address these issues, this paper proposes a novel Dynamic Efficient Load Balancing in Cloud using kookaburra Infused pelican Optimization for virtUal Server (DELICIOUS) is developed for effective load balancing process in cloud computing environment. The Hybrid Kookaburra-Pelican Optimization Algorithm (HK-POA) is implemented for offloading decisions which optimizes resource allocation and enhances user experiences. The evaluation of the performance of the DELICIOUS framework involves a thorough assessment that includes essential metrics such as throughput, execution time, latency, waiting time, computational complexity, and computational cost. The simulation experiments of the proposed DELICIOUS framework are conducted using CloudSim and achieves a better throughput of 1206.6 Kbps whereas, the GRAF, QoDA-LB, and RATS-HM technique attains 865 Kbps, 943.4 Kbps, and 984.6 Kbps respectively for intelligent load balancing in cloud networks.

Keywords *– Load Balancing, Cloud Computing, Kookaburra Optimization Algorithm, Pelican Optimization Algorithm.*

1. INTRODUCTION

Cloud Computing enables the efficient utilization of computing resources through its dynamic service model, requiring adaptive resource allocation and scalability to ensure Quality-of-Service (QoS) while minimizing resource usage [1, 2]. These resources cover computing power, storage, databases, networking, planning, resource finding, security, and privacy. Load balancing involves distributing workload effectively across multiple computing platforms, aiming to optimize system output, resource utilization, and VM performance metrics. Various load-balancing algorithms are employed by the cloud system to achieve resource efficiency [3, 4].

Load balancing algorithms varies depending on the system's condition which distinguishes static and dynamic methods. Static algorithms rely on heuristics and are contingent on the current system state, while dynamic algorithms utilize metaheuristics and operate independently of specific conditions [5, 6]. Dynamic algorithms are particularly effective in environments where the volume of requests and VMs varies significantly. These algorithms perform better in redistributing workloads among VMs to rectify load imbalances [7, 8].

Load refers to the tasks allocated to VMs, which may lead to imbalances due to underutilization or overutilization. Overutilization happens when tasks allocated to a VM exceed its capacity threshold, whereas underutilization occurs when a VM can handle more tasks than it currently hosts [9-10]. Therefore, load balancing is crucial for maintaining equilibrium among cloud resources. Load balancing and task scheduling are both required to accomplish this balance and ensure equitable allocation among virtual machines [11].

The load balancing and task scheduling is essential for meeting QoS requirements which falls into the category of NP-hard problems due to the multitude of scheduling and balancing parameters. When a single VM becomes overloaded while numerous empty VMs exist within the cloud network, redistributing workloads from overloaded to underutilized VMs is advantageous [12]. In cloud environments, it might be challenging to calculate all possible mappings of task resources, and even more to identify optimal mapping. Therefore, an effective task

distribution method is needed to schedule tasks in a way that prevents VMs from becoming excessively overloaded or underloaded [13]. The major contributions of the proposed DELICIOUS approach are as follows,

- This research proposes a novel, Dynamic Efficient Load Balancing in Cloud using kookaburra Infused pelican Optimization for virtUal Server (DELICIOUS) framework is to provide high quality services to customers in cloud computing applications.
- Initially, a task scheduling process is implemented to assign deadlines and execution times to tasks, while a load balancing process ensures workload migration in case of VM violations, due to that maintains load balance in the cloud environment.
- The performance evaluation of the DELICIOUS framework encompasses a comprehensive assessment, focusing on key metrics such as throughput, execution time, latency, waiting time, computational complexity, and computational cost.

The remainder of the study is organized as follows. Section II contains related papers with updates on recent research and descriptions of load balancing and task scheduling. Additional details about the proposed framework are provided in Section III. Section IV presents the experimental results of the proposed framework. The conclusions and further research are presented in Section V.

2. LITERATURE SURVEY

This section contains a summary of the literature has been discussed in this research. The concept of load balancing is initially discussed, along with its established model, metrics, and algorithms. Subsequently, it discusses recent literature on Load Balancing, presenting suggested algorithms by researchers and comparing their proposals with existing algorithms in the field.

In 2020, Devaraj, A.F.S., et al. [14] suggested a load balancing method with Firefly and Advanced Multi-Objective Particle Ensemble Optimization (FIMPSO). The simulation results revealed that the FIMPSO algorithm achieved the most efficient outcomes, with a shortest common response time of 13.58ms, surpassing all other comparable techniques. It provides the highest CPU utilization of 98%, memory utilization of 93%, dependability rating of 67%, throughput of 72%, and maximum make span of 148% respectively.

In 2021, Park, J., et al. [15] suggested GRAF, a proactive resource allocation technique utilizing graph neural networks to minimize overall CPU usage while assigning latency Service Level Objectives (SLOs). In comparison to autoscaling approaches, GRAF can achieve SLO latency with up to 19% of CPU consumption, according to experiments conducted using a number of publicly available benchmarks. Additionally, GRAF efficiently handles traffic spikes with 36% fewer resources than Kubernetes autoscaling and achieves faster latency convergence, up to 2.6 times respectively.

In 2022, Latchoumi, T.P., and Parthiban, L. [16] suggested to provide efficient resource scheduling in Cloud Computing (CC) scenarios with an essentially Quasi Opposite Dragonfly Algorithm technique for Load Balancing (QODA-LB). QODA-LB aimed to reduce task execution costs and times while ensuring an even distribution of workload across all VMs in the CC system. Simulation results demonstrated superior performance over leading methods, achieving optimal load balancing efficiency.

In 2022, Bal, P.K., et al. [17] suggested RATS-HM, a hybrid machine learning approach that blends safe resource allocation in a cloud computing context with effective task scheduling. Through simulations across different setups, RATS-HM shows the effectiveness which is compared to other state-of-the-art methods.

In 2023, Al Reshan, M.S., et al. [18] suggested an approach to load balancing in cloud computing which is fast and globally optimal. The suggested approach fused Gray Wolf Optimization with Particle Swarm Optimization (GWO-PSO) to achieve the advantages of both global optimization and quick convergence. By utilizing the GWO-PSO algorithm, this technique improves PSO convergence to 97.253% while reducing 12% on overall response times.

In 2023, Ramya, K., and Ayothi, S., [19] suggested Hybrid Whale and Dingo Optimization Algorithm (HDWOA-LBM) approach for cloud computing environments. This approach imitates a dingo's hunting behavior, optimizes the assignment of tasks to the appropriate virtual computer. The simulation experiments of HDWOA-LBM achieve a significant improvement in throughput of 21.28%, reliability of 25.42%, make span of 22.98%, and resource allocation of 20.86% for intelligent load balancing.

In 2024, Khaleel, M.I., [20] suggested RASA and dynamic task scheduling approach to balance the load in a cloud computing system. The RASA approach categorizes the tasks based on critical parameters using a task classification model. This approach resulted in reductions in latency overhead of 9%, processing time of 14%, workload imbalance of 15%, energy consumption of 19%, and idle time of 26%, along with improvements in resource availability of 22%, resource efficiency of 27%, and throughput of 32% respectively.

In order to address the above issues with cloud platforms, this research offers an agile task scheduling strategy that provides priority to crucial task characteristics including deadlines and durations which significantly affects the Quality of Service (QoS). Through meticulous scheduling and adherence to VM constraints, the algorithm ensures a well-balanced workload distribution across the cloud infrastructure.

3. DYNAMIC EFFICIENT LOAD BALANCING IN CLOUD USING KOOKABURRA INFUSED PELICAN OPTIMIZATION FOR VIRTUAL SERVER

In this section, a novel Dynamic Efficient Load Balancing in Cloud using Kookaburra-Infused Pelican

Optimization for Virtual Server (DELICIOUS) framework is proposed to deliver high-quality services to clients in Cloud Computing applications. The proposed DELICIOUS approach performs two main processes such as the task scheduling process, which is responsible for assigning deadlines and completion times to tasks, and the load balancing process, which manages the migration of workloads within Virtual Machine (VM) breach case to maintain load balance in cloud environment. Initially, the tasks requested by the user will be fed into the data center controller. These tasks are then passed to the load balancer, which uses a learning agent based on the Kookaburra-Pelican Hybrid Optimization Algorithm (HK-POA) to determine appropriate values and allocate tasks for machines in virtual environments suitable for cloud computing environments. The VM Manager then schedules tasks on the VM based on workload conditions such as underload and overload identified in the environment, using the migration process. The general block diagram of the proposed DELICIOUS framework is illustrated in Figure 1.

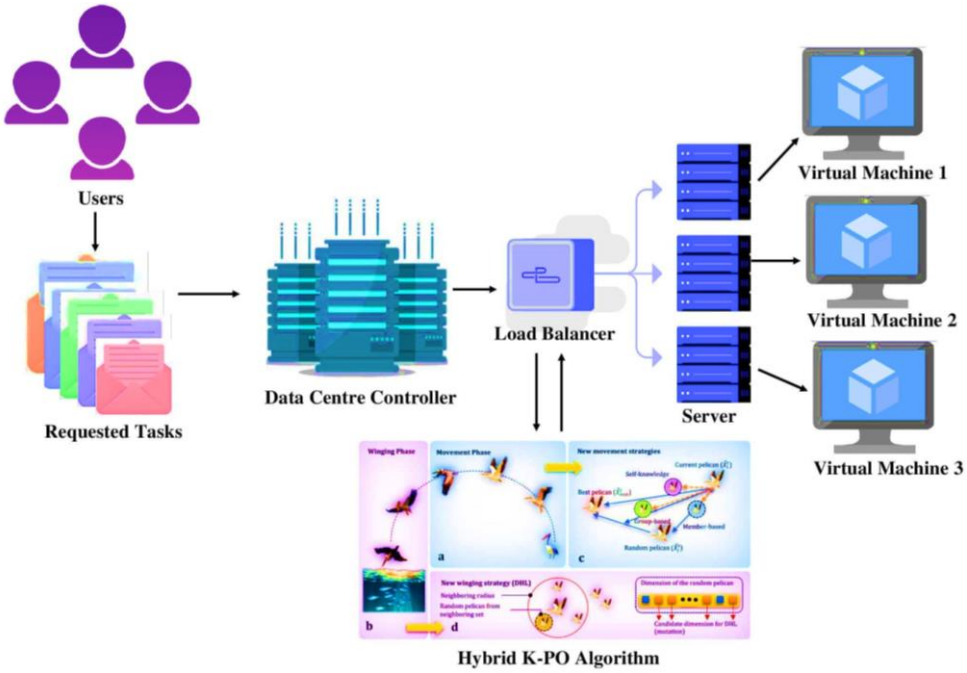

Figure 1. Proposed DELICIOUS Framework

3.1. Task classification

In task classification, algorithm 1 describes the process of adding tasks to the queue, with the primary determining factor being the task deadline. The underlying cloud hardware can accommodate a predetermined number (v) of virtual machines (VMs) to host tasks that users submit to the cloud scheduler. Dispatching these tasks presents challenges due to parameters such as τ_{cpu}, τ_{mem}, τ_{eed}, τ_{ded}, and τ_{ic} characterizing each task, alongside each VM being associated with ν_{cpu} and ν_{mem}, denoting CPU and memory resources, respectively. Each task demands a portion of the VMs' finite resources, contingent upon its size and duration. In response to this requirement, incoming workloads are classified into three distinct queues which is ϱ_{cpu}, ϱ_{mem}, and ϱ_{io}. For example, ϱ_{cpu} encompasses tasks necessitating intensive CPU utilization, ϱ_{mem} comprises those requiring substantial memory utilization, and ϱ_{io} contains tasks with prolonged durations.

Based on its parameters, each tasks speed is determined using R=(τ/S). The task category with the greatest rate is chosen as τC=max [R_{cpu}, R_{mem}, R_{io}]. After being accepted, tasks are initiated and categorized into three groups which is τC_{mem}, τC_{cpu}, and τC_{io}.. Then, the outer loop makes sure that each authorized task is finished on schedule. The execution time of each task is calculated, ensuring adherence to its Service Level Agreement (SLA), including its deadline. A queue, ϱ_{queue}, is employed to organize all accepted tasks meeting their deadlines. The loop verifies that tasks are arranged suitably according to their requirements by continuously assessing the memory, CPU, and I/O speeds of each task. Tasks demanding significant CPU utilization are classified first, followed by those necessitating higher memory utilization, and finally, those requiring longer durations are separated.

3.2. Load Balancing Via Hybrid K-POA

This section commences by elucidating the inspiration and theoretical foundation behind the proposed Hybrid Kookaburra-Pelican Optimization Algorithm (HK-POA), followed by a mathematical modeling of its implementation steps for solving optimization problems. Kookaburra optimization is an iterative method for solving problems in the realm of problem solving that uses stochastic search to produce effective solutions to optimization problems. Based on the collective population matrix of Kookaburras, the Kookaburra Optimization Algorithm (KOA) is represented mathematically by equation (1). According to equation (2), kookaburras' starting locations during KOA deployment are chosen at random.

$$X = \begin{bmatrix} X_1 \\ . \\ . \\ . \\ X_i \\ . \\ . \\ . \\ X_N \end{bmatrix}_{N\times m} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,1} & \cdots & x_{i,d} & \cdots & x_{i,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,d} & \cdots & x_{N,m} \end{bmatrix}_{N\times m} \quad (1)$$

$$x_{i,d} = lb_d + r(ub_d - lb_d) \quad (2)$$

The general KOA matrix, represented by X includes the search space. N is the total number of kookaburras, with m optimal variables. A random number inside the interval [0, 1] is shown, and the variables ub_d and lb_d, respectively, indicate the lower and upper bounds of the ith decision variable through the variable r. Equation (3) demonstrates that a vector can be used to describe the collection of objective function values that were assessed for the task.

$$F = \begin{bmatrix} F_1 \\ . \\ . \\ . \\ F_i \\ . \\ . \\ . \\ F_N \end{bmatrix}_{N\times 1} = \begin{bmatrix} F(X_1) \\ . \\ . \\ . \\ F(X_i) \\ . \\ . \\ . \\ F(X_N) \end{bmatrix}_{N\times 1} \quad (3)$$

The vector F, where F_i identifies the objective function evaluated corresponding to the ith kookaburra, specifies the collection of objective functions assessed in this scenario. The prey set that each kookaburra has access to is then ascertained by comparing the values of the objective functions, as indicated by equation (4).

$$CP_i = \{X_k : F_k < F_i and\ k \neq i\}, where\ i = 1, 2, \ldots\ldots, N\ and\ k \in \{1, 2, \ldots\ldots, N\} \quad (4)$$

In this case, the set of possible prey that the ith kookaburra can approach is shown by CP_i. F_k represents the objective function value in this instance, and X_k represents the kookaburra with an objective function value greater than the ith kookaburra. Using equation (5), which mimics the kookaburra's path toward the prey while hunting, the bird's new location is ascertained. The kookaburra in concern will relocate to this new site if the objective function's value there rises, as demonstrated by equation (6).

$$x_{i,d}^{p1} = x_{i,d} + r\left(SCP_{i,d} - I.x_{i,d}\right), i = 1, 2, \ldots\ldots, N, \quad and\ d = 1, 2, \ldots\ldots, m \quad (5)$$

$$X_i = \begin{cases} x_i^{p1}, & F_i^{p1} < F_i \\ X_i, else \end{cases} \quad (6)$$

F_i^{p1} represents the functional objective value, $x_{i,d}^{p1}$ represents the new recommended position of the ith kookaburra based on the first step of the KOA and $x_{i,d}^{p1}$ represents the kookaburra's dimension d. Furthermore, r is a random number selected from 0 to 1 based on a normal distribution. The $SCP_{i,d}$ represents the d-th dimension of the prey chosen for the ith kookaburra, where i is a randomly chosen number from the set {1, 2}. N stands for the total number of kookaburras, and m for the number of decision variables.

The KOA technique imitates the behaviour of kookaburras around hunting sites by utilizing equation (7) to determine an arbitrary location. Equation (8) indicates that the previous position will be replaced if the new location for each kookaburra raises the value of the objective function.

$$x_{i,d}^{p2} = x_{i,d} + (1 - 2r).\frac{(ub_d - lb_d)}{t}, i = 1, 2, \ldots\ldots, N, d = 1, 2, \ldots\ldots, m, and\ t = 1, 2, \ldots\ldots, T \quad (7)$$

$$X_i = \begin{cases} x_i^{p2}, & F_i^{p2} < F_i \\ X_i, else \end{cases} \quad (8)$$

The function's target value in this instance is F_i^{p2}, the dth dimension is represented by x_i^{p2}, and the ith kookaburra's proposed new location is x_i^{p2}, which is based on the second phase of KOA. Furthermore, T denotes the algorithm's maximum iteration count, and t denotes the algorithm's iteration counter.

The proposed PO algorithm is a population-based approach where pelicans constitute the members of this population. Initially, the extracted features are randomly initialized within the lower and upper bounds of the problem using Equation (9).

$$M_{p,q} = h_q + rand \cdot \left(k_q - h_q\right), p = 1, 2, \ldots, X, q = 1, 2, \ldots y \quad (9)$$

This equation gives the value of the variable qth, which is decided by the candidate solution pth, represented by $M_{p,q}$. The lower and upper bounds, represented by h_q and k_q, respectively, express the total population. The mathematical expression for the pelican's strategy as it moves towards its prey is provided in equation (10).

$$M_{p,q}^{A_1} = \begin{cases} M_{p,q} + rand \cdot \left(A_q - G \cdot M_{p,q}\right), & E_a < E_p \\ M_{p,q} + rand \cdot \left(M_{p,q} - A_q\right), & else \end{cases} \quad (10)$$

This equation defines $M_{p,q}^{A_1}$, indicating the updated status of the pth pelican in the qth dimension after phase 1. The variable G takes on a random value of either 1 or 2. A_q represents the prey's position in the qth dimension, while E_a denotes its objective function value. Hence, parameter G significantly influences PO's exploration by accurately selecting features. This behavior, reminiscent of pelicans during hunting, is mathematically simulated in the equation (11).

$$M_{p,q}^{A_2} = M_{p,q} + D \cdot \left(1 - \frac{t}{T}\right) \cdot (2 \cdot rand - 1) \cdot M_{p,q} \quad (11)$$

In this equation, $M_{p,q}^{A_2}$ represents the updated state of the pth pelican in the qth dimension after the time frame. D(1-t/T) gives the neighbourhood radius of $M_{p,q}$ with D fixed at 0.2. The coefficient D(1-t/T) is pivotal for PO's exploitability, as it brings the algorithm closer to the global optimal solution. Initially, during the early iterations, this coefficient has a larger value, resulting in a broader

exploration around each member. However, as the algorithm progresses through subsequent phases, the pelican positions are adjusted, and the most promising features are identified. Eventually, the algorithm stops and produces the best features when the maximum number of iterations is reached or the convergence condition is satisfied. The algorithm 1 shows the steps for HK-POA which is given below.

Initialize Kookburra population X_k and pelican population X_p.

Initialize iteration count $t = 0$.

Exploration Phase of Kookaburra

For each Kookaburra individual x_{ki} in X_k:

Update the position using:

$$x_{ki} = x_{ki} + \alpha.rand(0,1).StepSize$$

Exploitation phase of Pelican

For each Pelican individual x_{pi} in X_p:

Update the position using:

$$x_{pi} = x_{pi} + \beta.rand(0,1).\nabla f(x_{pi})$$

Combine Kookaburra and Pelican populations: $X = X_k \cup X_p$

Select individuals for the next generation based on fitness.

Increment t.

If termination conditions are not met stop;

Otherwise, go to step 3.

---

4. RESULTS AND DISCUSSION

In this section, the simulation setup and outcomes of the proposed DELICIOUS method employing various validation and assessment studies are discussed. In the experimental evaluation of the proposed DELICIOUS method, classes from the extended CloudSim toolkit are utilized for modelling and simulation of cloud systems. CloudSim Simulator allows to set up a virtualized environment with on-demand resource provisioning. Furthermore, cloud services and related applications may be more easily simulated, modelled, and tested. A few key factors taken into account are throughput, execution time, latency, waiting time, computational complexity, and computational cost which is then compared to the number of available tasks.

4.1. Performance Evaluation

The efficiency of the proposed DELICIOUS approach was assessed using critical performance indicators for the first time, including throughput, latency, execution time, delay, computational complexity, and computation time and fundamental methods of implementation, such as RATS-HM, GRAF, and QoDA-LB. Figure 2 displays the throughput attained for multiple tasks using the proposed DELICIOUS approach along with comparison methodologies. The proposed DELICIOUS design, performs better than the other three systems when processing workloads out of 500 tasks, with throughputs of 865 Kbps, 943.4 Kbps, and 984.6 Kbps respectively.

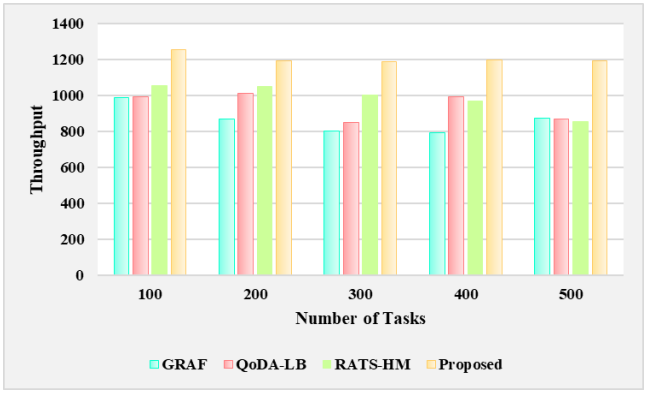

Figure 2. Throughput Vs Number of Tasks

These outcomes shows that the proposed DELICIOUS approach is effective and that it is feasible to take advantage of integrated HK-POA benefits. Through a comprehensive examination of numerous factors, this method delivers optimal throughput regardless of the workload entering the cloud environment by allocating incoming tasks to the right VM's.

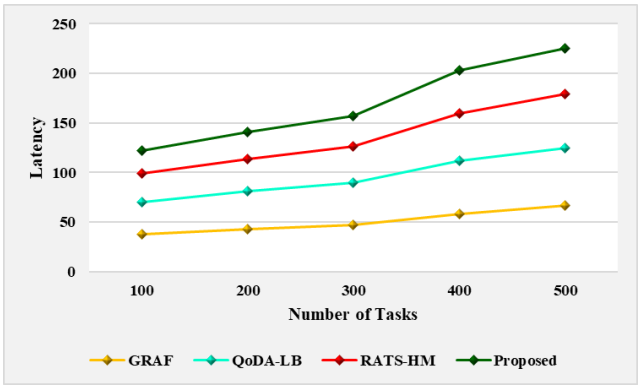

Figure 3. Latency Vs Number of Tasks

Figure 3 illustrates the delay that the proposed DELICIOUS approach, as well as the previously stated GRAF, QoDA-LB, and RATS-HM techniques, encountered for varying numbers of tasks. The performance of the proposed DELICIOUS method is superior to the methods discussed, as shown by the latency trends for varying task counts.

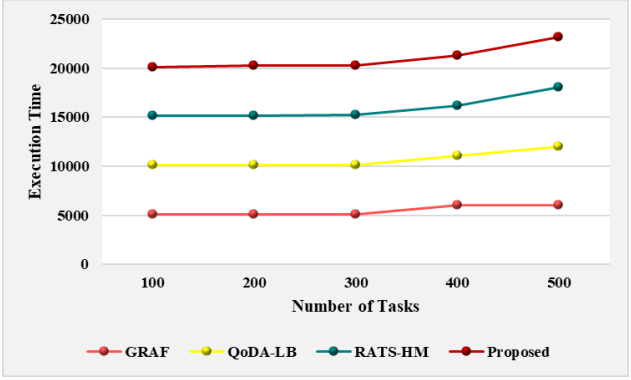

Figure 4. Execution Time Vs Number of Tasks

This decrease in latency is credited to the scheme's adeptness and consistency in dynamically applying diverse load balancing constraints during task scheduling. Moreover, it excels in effectively allocating tasks to suitable VMs, thus averting situations of under-utilization or over-utilization in the cloud environment.

Furthermore, Figure 4 and Figure 5 displays the proposed DELICIOUS method for various workloads together with the latency and execution durations for the evaluated GRAF, QoDA-LB, and RATS-HM approaches. Particularly, the proposed DELICIOUS scheme exhibits reduced execution time and waiting time relative to the baseline approaches under varying task.

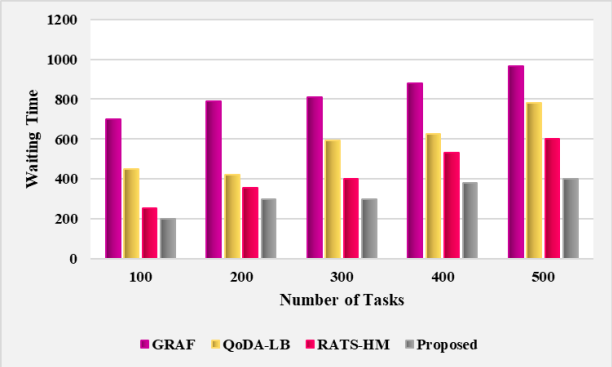

Figure 5. Waiting Time Vs Number of Tasks

By integrating the multi-dimensional HK-POA, the scheme explores essential factors for assigning incoming tasks to suitable VMs within the cloud environment. This dynamic exploration facilitated by the proposed DELICIOUS scheme effectively decreases task waiting times by dynamically adjusting the load balancing rate to meet the required criteria.

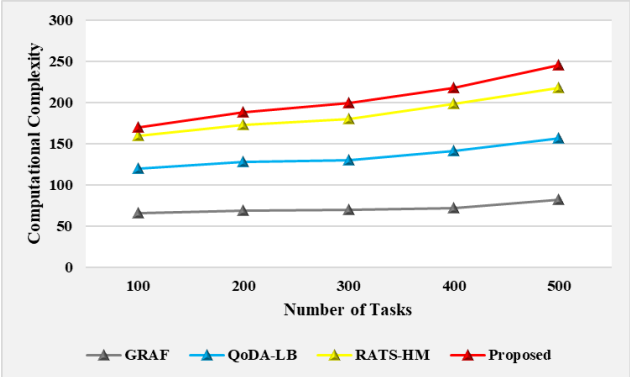

Figure 6. Computational Complexity Vs Number of Tasks

Figures 6 and 7 display the computational complexity, cost, and techniques of comparison of the suggested system for various activities. The efficacy of the proposed DELICIOUS system is assessed in this part. The basic load balancing techniques, such as the GRAF, QoDA-LB, and RATS-HM schemes, concentrate on task efficiency, computing complexity, and cost.

By including constraints and multi-objective optimization elements into the load balancing process, the proposed DELICIOUS strategy combines processing complexity and cost, independent of the number of operations. Moreover, it includes diverse load balancing parameters to dynamically assign tasks to different VMs throughout the allocation process.

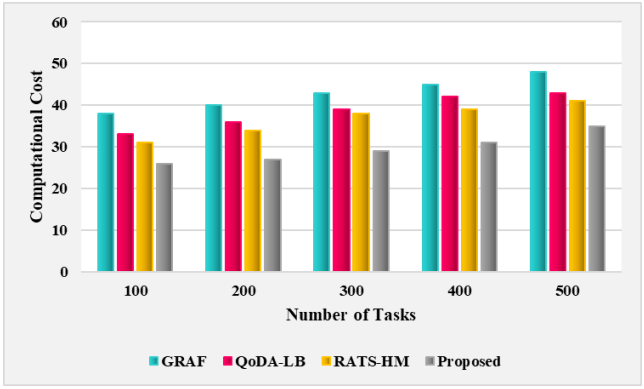

Figure 7. Computational Cost Vs Number of Tasks

5. CONCLUSION

This paper proposed a novel Dynamic Efficient Load Balancing in Cloud using kookaburra Infused pelican Optimization for virtUal Server (DELICIOUS) is developed for Effective load balancing system in cloud computing. The proposed DELICIOUS technique is validated by using the Cloud Simulator (CloudSim) to providing consumers with the best services for applications using cloud computing. Furthermore, the performance of the proposed DELICIOUS technique is evaluated in terms of the parameters such as throughput, execution time, latency, waiting time, computational complexity, and computational cost. The DELICIOUS technique achieves a better throughput of 1206.6 Kbps whereas, the GRAF, QoDA-LB, and RATS-HM technique attains 865 Kbps, 943.4 Kbps, and 984.6 Kbps respectively.

CONFLICTS OF INTEREST

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

FUNDING STATEMENT

Not applicable.

ACKNOWLEDGEMENTS

The author would like to express his heartfelt gratitude to the supervisor for his guidance and unwavering support during this research for his guidance and support.

REFERENCES

[1] F. Alqahtani, M. Amoon, and A.A. Nasr, “Reliable scheduling and load balancing for requests in cloud-fog computing”, *Peer-to-Peer Networking and Applications*, vol. 14, no. 4, pp.1905-1916, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[2] K. Balaji, P.S. Kiran, and M.S., Kumar, “An energy efficient load balancing on cloud computing using adaptive cat swarm optimization”, *Materials Today: Proceedings*, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[3] F.M. Talaat, M.S. Saraya, A.I. Saleh, H.A. Ali, and S.H. Ali, “A load balancing and optimization strategy (LBOS) using reinforcement learning in fog computing environment”, *Journal of Ambient Intelligence and Humanized Computing*,

vol. 11, no. 11, pp.4951-4966, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[4] S. Sagar, M. Ahmed, and M.Y. Husain, "Fuzzy Randomized Load Balancing for Cloud Computing", *In International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pp. 18-29, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[5] N. Arivazhagan, K. Somasundaram, D. Vijendra Babu, M. Gomathy Nayagam, R.M. Bommi, G.B. Mohammad, P.R. Kumar, Y. Natarajan, V.J. Arulkarthick, V.K. Shanmuganathan, and K. Srihari, "Cloud-internet of health things (IOHT) task scheduling using hybrid moth flame optimization with deep neural network algorithm for E healthcare systems", *Scientific Programming*, 2022, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[6] A. Asghari, and M.K. Sohrabi, "Combined use of coral reefs optimization and reinforcement learning for improving resource utilization and load balancing in cloud environments", *Computing*, vol. 103, no. 7, pp.1545-1567, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[7] H. Mahmoud, M. Thabet, M.H. Khafagy, and F.A. Omara, "An efficient load balancing technique for task scheduling in heterogeneous cloud environment", *Cluster Computing*, vol. 24, no. 4, pp. 3405-3419, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] Z. Miao, P. Yong, Y. Mei, Y. Quanjun, and X. Xu, "A discrete PSO-based static load balancing algorithm for distributed simulations in a cloud environment", *Future Generation Computer Systems*, vol. 115, pp.497-516, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[9] J.R. Adaikalaraj, "Load Balancing In Cloud Computing Environment Using Quasi Oppositional Dragonfly Algorithm", *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 10, pp. 3256-3273, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] S. Dhahbi, M. Berrima, and F.A. Al-Yarimi, "Load balancing in cloud computing using worst-fit bin-stretching", *Cluster Computing*, vol. 24, no. 4, pp. 2867-2881, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[11] S. Afzal, and G. Kavitha, "Load balancing in cloud computing–A hierarchical taxonomical classification", *Journal of Cloud Computing*, vol. 8, no. 1, pp. 22, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[12] U. K. Jena, P. K. Das, and M. R. Kabat, "Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment", *Journal of King Saud UniversityComputer and Information Sciences*, vol. 34, no. 6, pp.2332- 2342, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] E. H. Houssein, A. G. Gad, Y. M. Wazery and P. N. Suganthan, "Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends", *Swarm and Evolutionary Computation*, vol. 62, pp.100841, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] A.F.S. Devaraj, M. Elhoseny, S. Dhanasekaran, E.L. Lydia, and K. Shankar, "Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments", *Journal of Parallel and Distributed Computing*, vol. 142, pp.36-45, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[15] J. Park, B. Choi, C. Lee, and D. Han, GRAF: A graph neural network based proactive resource allocation framework for SLO-oriented microservices", *In Proceedings of the 17th International Conference on emerging Networking Experiments and Technologies*, pp. 154-167, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[16] T.P. Latchoumi, and L. Parthiban, "Quasi oppositional dragonfly algorithm for load balancing in cloud computing environment", *Wireless Personal Communications*, vol. 122, no. 3, pp.2639-2656, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[17] P.K. Bal, S.K. Mohapatra, T.K. Das, K. Srinivasan, and Y.C. Hu, "A joint resource allocation, security with efficient task scheduling in cloud computing using hybrid machine learning techniques", *Sensors*, vol. 22, no. 3, pp.1242, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[18] M.S. Al Reshan, D. Syed, N. Islam, A. Shaikh, M. Hamdi, M.A. Elmagzoub, G. Muhammad, and K.H. Talpur, "A fast converging and globally optimized approach for load balancing in cloud computing", *IEEE Access*, vol. 11, pp.11390-11404, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[19] K. Ramya, and S. Ayothi, "Hybrid dingo and whale optimization algorithm-based optimal load balancing for cloud computing environment", *Transactions on Emerging Telecommunications Technologies*, vol. 34, no. 5, p.e4760, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[20] 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", *Journal of Network and Computer Applications*, vol. 221, pp.103788, 2024. [CrossRef] [Google Scholar] [Publisher Link]

AUTHORS

G. Saranya received her B.E degree in Computer Science and Engineering from Anna University, Chennai and M.E degree in Computer Science and Engineering from Hindustan University, Chennai. She started her career as an Assistant Professor and has 9 years and 6 months of experience. Currently she is working as an Assistant Professor in S.A. Engineering College, Chennai. Her research interests include Deep Learning and Cloud Computing. She is a lifetime member of ISTE.

G Belshia Jebamalar currently working as Assistant professor in Department of computer science in SA Engineering college, Poonamallee, Thiruverkadu, Tamil Nadu 600077 India

Chukka Santhaiah I am currently associated with Sri Venkateswara College of Engineering, Tirupati. My current responsibilities include coordinating all the departmental operations, delivering the lectures on time, conducting the class assignments and carrying out necessary evaluation and organizing mid-sem exams, class assignments along with external and internal lab exams & evaluating the same. I am also responsible for conducting seminars, conferences, workshops. My academic credentials include a Doctor of Philosophy (Ph.D.) (CSE) from Sri Venkateswara University College of Engineering, India, Master of Technology (M.Tech) (CS) from Rajiv Gandhi College of Engineering, India, Bachelor of Technology (B. Tech) in CSE from Rajiv Gandhi College of Engineering, India. Proven record of13 +patents 3granted, and one text book published lambert publications Submitted research proposals for DST and SERB. Applied for IP award and DST Inspire award. Research Areas: Image Processing, Bioinformatics, Computer Networks, Machine Learning, Artificial Intelligence, and Data Science.

Arrived: 05.07.2024
Accepted: 10.08.2024