A Krill Herd Behaviour Inspired Load Balancing of Tasks in Cloud Computing

Raed Abdulkareem HASAN*, Muamer N MOHAMMED
Faculty of computer system and software engineering,
University Malaysia Pahang (UMP)
Kuantan-26300, Pahang, Malaysia
e-mail: raed.isc.sa@gmail.com (**Corresponding author*)
e-mail: muamer@ump.edu.my

Abstract: A developing trend in the IT environment is mobile cloud computing (MCC) with colossal infrastructural and resource requirements. In the cloud computing environment, load balancing - a way of distributing workloads across numerous computing resources, is a vital aspect. A proficient load balancing guarantees an effective resource usage through the supply of network resources based on the user demands. It can also organize the network clients using the fitting planning criteria. This paper sets forth an advanced load balancing and energy/cost aware technique for a demand-based network resource allocation in cloud computing. The load balancing process in the proposed strategy utilizes a Krill load balancer (Krill LB) which is expected to achieve a well-balanced load over virtual machines. The aim of using the Krill LB as the load balancer is to increase the throughput of the network as much as possible. The speed, task cost, and weight of the tasks were first determined, after which, the Krill herd optimization algorithm was for the load balancing based on the measured parameters. Furthermore, a modified dynamic energy-aware cloudlet-based mobile cloud computing model (MDECM) was introduced for energy cost awareness in load balancing based on the service rate and energy of the mobile users. The proposed work was aimed at optimizing resource allocation in MCC in an energy-efficient manner. The performance of the suggested Krill-LB was benchmarked against that of Honey Bee Behavior Load Balancing (HBB-LB), Kill Herd, and Round Robin algorithms.

Keywords: Mobile cloud computing, Resource allocation, Optimization, Load balancing, Energy cost aware.

1. Introduction

Mobile cloud computing (MCC) is characterized as the augmentation of cloud computing with another ad-hoc infrastructure for mobile gadgets [7]. MCC is a promising framework that brings effective cloud computing into a versatile processing condition, in which mobile devices interface with the internet via a wireless network, and afterward, associate with the remote cloud [18]. MCC at its easiest, alludes to an infrastructure where data storage and processing occur exteriorly to mobile gadgets [10]. This is a rising cloud benefit model which is succeeding the pattern to spread out the cloud to the brink of systems [28]. In addition, it encourages the building of smart mobile gadgets with improved cloud accessibility [13]. This is an internet-based generation where mobile devices are loaded with many applications. The context is distinctively contrasting with mobile computing since in MCC, gadgets run cloud-centered web applications whereas mobile computing runs with apps [27]. With the fast growth of mobile applications accompanied by cloud computing, MCC as a promising technology aimed at mobile services is drawing more attention [22]. The primary components of MCC include the migration of computationally intensive tasks or applications to servers in cloud domain in order to execute them, and after that, recover the outcome of the execution from these servers [8]. It utilizes communication technology to share information and assets and incorporates location-aware technologies, mobile access to IT, and energy sparing technology specifically designed for mobile devices [15]. As of late, mobile applications have been noticeably copious with different classes, such as entertainment, health, games, business, social networking, travel, and news [21][5][24].

In the cloud, the load assigned to every node in the network is similarly distributed with an even quantity of resources over time. This enhances the scheme performance by moving the workloads among various nodes [4][26]. The primary goal is to expedite the implementation of applications on resources whose workload changes at the runtime in an unpredictable way. These are generally discussed in homogeneous conditions such as grids. Fundamentally, there are two ways of load balancing procedures: (i) Static and (ii) dynamic [3]. The load balancing plan and a migration policy are aimed at virtual machine (VM) clustering to brilliantly choose a VM from an over-burdened

https://doi.org/10.24846/v26i4y201705

resource and transfer it elsewhere where the load is less [25]. A load-balancing heuristic mechanism ensures the offloading of resources to the cloud, with the aim that the balance between the mobile device and cloud is augmented [14]. Load balancing in cloud ought to be dispersed, flexible and extensible. Albeit, many works have been done in various aspects of a distributed scheme on the issue of load balancing [2]. Load balancing, along with high resource usage, is accomplished by getting all the physical resources used, and after that, initializes the VM movement based on a pre-determined approach [20][1].

In cloud computing, load balancing or resource control is a major challenge. Load balancing is a procedure of workload distribution over various PCs or different resources over the system link to accomplish an ideal resource use, expand throughput, reduce response time, and keep away from over-burden [6]. To take a shot at load balancing, conventional algorithms like First Agent-Based Dynamic Load Balancing in Cloud Computing (ABDLB) [11], Load Balancing Ant Colony Optimization (LBACO) [16], First Come First Serve (FCFS), Round Robin (R), Random Allocation (R), Shortest Job First (SJF) and Longest Job First (LJF) are not adequate, and the meta-heuristic algorithms such as Evolutionary Algorithms and Swarm Intelligent Algorithms have been investigated [12][23]. The algorithm presented in this study is an efficient load balancing technique believed to be suitable for application in a cloud computing environment. This study contributes to the existing body of knowledge on load balancing in the following ways:

- Developing a Krill Herd behavior-inspired algorithm for an effective scheduling and balancing of non-preemptive independent tasks in cloud computing environments.
- Surveying the merits and demerits of the existing load balancing algorithms.
- Correlating the suggested Krill-LB algorithm with actual foraging behavior of Krill Herd using a clear flow diagram of the behavioral control structures of Krill herd and Krill-LB.
- Performing a systematic and analytical study with mathematical evidence of the performance of the proposed algorithm in a cloud computing environment.
- Benchmarking the performance of the proposed algorithm with that of the existing load balancing algorithms.

The current paper is organized as follows: Section 1 presents the study background and the problem to be addressed, while Section 2 presents a review of the related studies, followed by a discussion of the strengths and weaknesses of the existing protocols. Section 3 introduces the proposed algorithm, the experimental steps, algorithm pseudocode, diagrams and mathematical equations, while Section 4 discusses the results of the experimental studies which have been compared to the existing load balancing algorithms. The last section concludes the study.

2. Related Works

Yanmin Gong et al. [9] have studied the privacy issues in the ad-hoc mobile cloud computing and suggested a structure that can ensure area privacy while assigning works to mobile devices. Their mechanism centered on differential privacy along with geocast. In addition, it enables mobile devices to accord their resources to the ad-hoc mobile cloud that is deprived of leaking their location details. They created analytical models and task allocation methodologies that balance privacy, utility and framework overhead in an ad-hoc mobile cloud. They additionally performed several experiments based on real-world datasets, and the outcomes demonstrated that this structure can secure the location privacy of mobile gadgets while furnishing compelling administrations with low framework overhead.

Kezhi Wang et al. [29] in C-RAN had suggested a joint energy minimization, together with resource allocation accompanied by MCC below the time restriction of the provided undertakings. They initially evaluated the energy and time model of the computation and communication. At that point, they have formulated the joint energy minimization into a non-convex optimization, accompanied by the limitation of transmitting power, computation capacity, a task executing time and also front haul information rates. They non-convex optimization was afterward, redeveloped into an identical convex issue in light of weighted minimum mean square error (WMMSE). In C-RAN accompanied by mobile cloud, an iterative algorithm was at last assigned to manage joint resource allocation. The simulation results affirmed that the suggested energy minimization and resource allotment solution can enhance the framework performance and save energy.

Changsheng You et al. [30] suggested a resource allotment aimed at a multiuser MECO framework

based on time-division multiple access (TDMA) accompanied by orthogonal frequency-division multiple access (OFDMA). To begin with, the ideal resource allocation intended for the TDMA MECO framework with infinite or finite cloud computation capacity was framed as a convex optimization issue for reducing the weighted totality mobile energy consumption under the limitation of computation latency. The ideal strategy was demonstrated to require a threshold-based structure with regard to an inferred offloading priority function that provides priorities for clients as indicated by their channel profits along with local computing energy consumption. Accordingly, clients with needs higher or below a given threshold separately perform a complete and least offloading. In addition, a sub-optimal resource-allocation algorithm meant for the cloud with limited capacity was suggested to lessen the multifaceted nature of computation for calculating the threshold. Subsequently, they considered that the OFDMA MECO framework, aimed at an ideal resource allotment as a mixed-integer issue.

Yanchen Liu et al. [19] suggested an Adaptive Multi-Resource Allocation, aimed at cloudlet-based MCC scheme. The suggested multi-resource allocation plan improves the mobile cloud service quality in terms of the framework throughput and service latency. They formulated the resource allocation model as a semi-Markov decision procedure below the average cost norm and additionally tackled the optimization issue using linear programming techniques. An ideal resource allocation guiding principle has been ascertained through maximizing the long-standing compensate while meeting the framework necessities of the demand blocking probability and service time latency. From the simulation outcome, it was demonstrated that the framework adaptively balanced the allocation policy on how much resources to be assigned and whether to utilize the distant cloud in line with the traffic of requested mobile services and the accessibility of resources in the framework. Their algorithm outperformed the greedy admission control on a broad range of environment.

Hongbin Liang et al. [17] suggested a service decision-making system meant for interdomain service exchange to adjust the computation loads among different cloud domains. Their framework concentrates on augmenting the prizes for both the cloud framework and the clients by limiting the extent of service dismissals that debase the client fulfillment level fundamentally. They figured the service entreaty decision-making procedure as a semi-Markov decision method. The ideal service transfer decisions were achieved by jointly considering the framework incomes and costs. Our system is an extensive simulation which demonstrated that the suggested decision-making procedure enhanced the framework outcomes and lessened service disturbances compared to the insatiable approach [6]. Finally, the calculation of energy-cost aware for an improved load balancing was performed and the demand aimed at the most minimal energy was done at cloudlet mediator.

3. Krill Load Balancing and Dynamic Energy-Aware in Mobile Cloud Computing

This paper suggested an optimized load balancing along with a modified dynamic energy aware cloudlet model in resource allocation. The Krill herd optimization algorithm was employed to augment load balancing; centered on the successful parameters such as speed, task cost, and weight. After the load balancing, the modified dynamic energy-aware cloudlet-based mobile cloud computing model (MDECM) was shown to reduce the service rate and energy of mobile devices, thereby achieving the intended energy cost aware and an effective resource allocation. The block diagram of the suggested framework is shown in Figure 1.

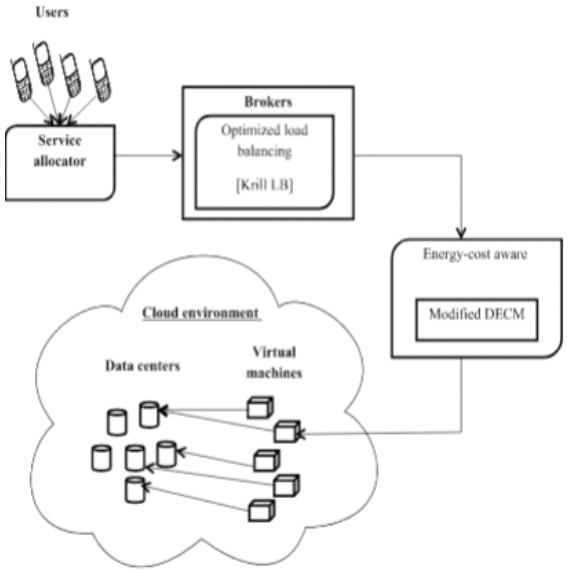

Figure 1. Block diagram of the suggested system

3.1 Service allocator

Initially, the task demands from all the mobile customers were directed to the cloud domain. The task demands were first collected from all the clients at a remote site signified as $R = \{r_1, r_2, r_3, ...r_n\}$, where R is the arrangement of the task demands from the mobile clients.

The task demands were gathered from the users at each time by the service allocator who considers both the tasks and the service. The service allocator designates the service for each demand. In the suggested work, an optimized load balancing, together with a modified DECM were used for effective resource allocation. These are further presented in the following subsections.

3.2 Krill herd-inspired load balancing (KH-LB)

In MCC, load balancing is employed for an optimized resource usage, maximize throughput, lessen response time, and evade over-burden of any single resource. Load balancing enhances the dissemination of workloads over many computing resources. In clouds, load balancing is a separate task that takes place in the VM as an imperative part of task scheduling. The load must be balanced at any time some VMs are laden to achieve an optimal machine use. VMs are under-stocked with tasks for processing. To reduce the issue of energy loss during load balancing, the proposed work utilized a Krill herd optimized load adjusting technique. Load balancing procedures solve the issue of load irregularity amongst VMs and are effective in decreasing the influence of span and response time.

The Krill herd load balancing (KH-LB) is a dynamic procedure which balances the load and nonetheless, considers the task priorities in the holding up lines of the VMs. The given algorithm is an augmentation of the existing dynamic load balancing approach with the incorporation of the Krill herd behavior concept. The accompanying parameters such as speed and task cost, as well as the weight of each task, were used in the Krill herd optimization algorithm to enhance the performance.

3.2.1 Speed

For a better usage of the cloud, the task speed needs to be expanded effectively. The mathematical formula for ascertaining task requests for speed is given as:

$$L = R_{\alpha} \big/ \beta, \tag{1}$$

where L is the task request for speed, R_{α} is the turnaround time of undertaking the demand, and β is the holding up time of task request for (R).

3.2.2 Task cost

The cost should be paid early to ensure the demand is met. The demand cost is determined using the equation:

$$C = \mu * \hat{a}, \tag{2}$$

where C is the task cost, μ is data rate of task demands, and $\hat{a}$ is the holding up time of task demands.

3.2.3 Weight

The amount of weight hinge on the speed and cost of demands. The weight is given by the equation:

$$W = \alpha * (L + C), \tag{3}$$

where W is the weight, L is the task request speed, C is the task cost, α is a constant value $\in [0, 1]$.

The pseudocode of the suggested Krill herd optimized load balancing algorithm, using speed, task cost and weight as parameters is shown in Figure 2.

Begin
(i) Define population size (S) and iteration (Imax)
(ii) Random initialization.
Set the iteration counter I = 1;
Initialize the population (***R***);
Set the foraging speed Vf, the maximum diffusion speed Dmax, and the maximum induced speed Nmax.
(iii) Fitness evaluation.
Evaluate each krill individual according to speed (***L***), task cost (***C***), weight (***W***).
(iv) **While** I <Imax do
Sort the population/krill from best to worst.
for i = 1: S (all krill) do
Perform the following motion calculation.
Movement induced by other krill individuals
Foraging activity
Physical diffusion
Implement the genetic operators.
Update the krill individual position in the search space.
Evaluate each krill individual according to its position.
End for i
Sort the population/krill from best to worst and find the current best.
Imax = I+1.
End while
(vi): Evaluate the krill best solution.
End

Figure 2. Pseudocode of the suggested Krill herd algorithm

3.2.4 Description of Krill Herd- Load Balancing (KH-LB) Algorithm

Step 1

The algorithm commences with the haphazard initialization of the task demand size (R).

Step 2

The fitness value is assessed from every Krill individual as indicated by the speed (L), task cost (C), and weight (W).

Step 3

Next, the fundamental loop of the algorithm begins by first arranging the Krill from the best to the utmost noticeable bad individual.

Step 4

The movement updates (induced movement, foraging, random diffusion) are computed for all the Krill using the following equations:

a) Foraging motion update

The foraging update is computed thus:

$$F_x(t+1) = V_f \beta_x + \omega_f F_x(t) \tag{4}$$

$$\beta_x = \beta_x^{food} + \beta_x^{best}, \tag{5}$$

where V_f is the foraging speed, ω_f is the inertia weight, β_x^{best} is the finest solution of the x^{th} Krill.

b) Induced movement update

The induced movement shows the level of maintenance of the Krill herd, with the maintenance given to each Krill given by:

$$M_x(t+1) = M_{\max}\alpha_x + \omega_n + M_x(t) \tag{6}$$

$$\alpha_x = \alpha_x^{\text{total}} + \alpha_x^{\text{target}} \tag{7}$$

where $M_{\max}$ is the maximum instigated speed, ω_n is the inertia weight, α_x^{total} is the local impact x^{th} Krill has on its neighbors, α_x^{target} is the best arrangement of the x^{th} Krill.

c) Physical diffusion update

The third motion update emulates the physical dissemination by random action, and is given as:

$$D_x(t+1) = D_{\max}\left(\frac{1-i}{i_{\max}}\right)\delta, \tag{8}$$

where $D_{\max}$ is the maximum diffusion speed, (δ) is the random directional vector in [-1, 1].

Step 5

This step is centered on the three already specified movements using the distinctive parameter of motion (time) and the location of x^{th} Krill in the interim to $t'+\Delta t'$ which is determined by Equation 9 and used to compute the position of each Krill.

$$K_x(t'+\Delta t') = K_x(t') + \Delta t' \frac{dK_x}{dt}, \tag{9}$$

where $\Delta t'$ is a standout among the utmost noteworthy constants and is better fine-tuned as far as the given real-world optimization. The location of an individual Krill in the tree is refreshed using the preceding equation to measure the objective function of the individual Krill towards the end of the algorithm where the best Krill (solution) re-occurred.

Step 6

Toward the end, the stopping criterion is used for the fulfillment of the predefined number of function assessments. When the stopping criterion is not met, sort the Krill population from the best to the least after that, compute the motion updates for all Krill and assess their positions. It restores the best solution (Krill) when the condition is met. These outcomes are the optimized load balance to the tasks. The whole process of the suggested Krill herd load balancing (KH-LB) is shown in Figure 3.

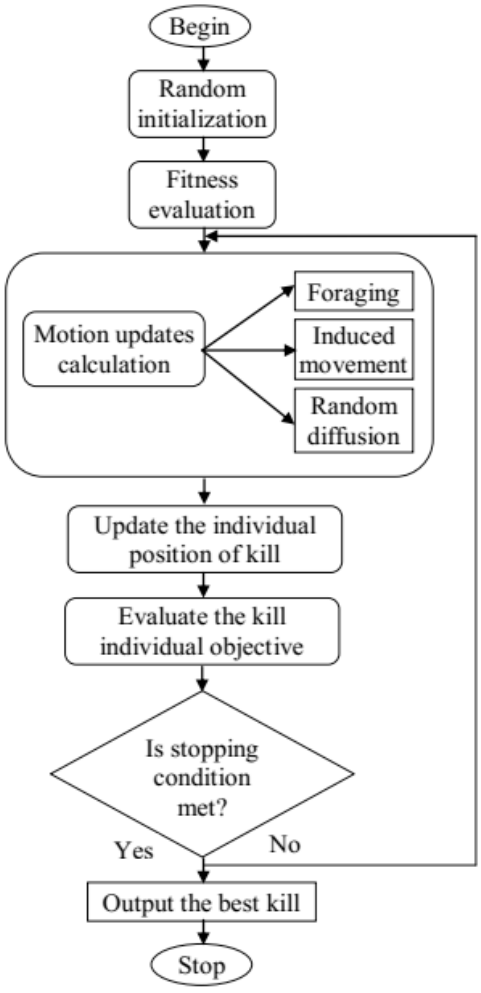

Figure 3. Krill herd load balancing algorithm

The optimized load balancing achieved with the Krill herd algorithm overcame the complexities in load balancing. After load balancing, the modified dynamic energy aware cloudlet model (MDECM) solved the energy loss issues and provided a great service in resource allocations.

3.3 Modified Dynamic Energy Aware Cloudlet Model (MDCEM)

The allocation of resources in MCC using dynamic energy aware model was performed using the MDECM model. The model was built using parameters, such as the service rate and the energy of each task. In the cloud, the customers or their representatives submit service demands from anywhere around the world. It is essential to see that there is a contrast between cloud consumers and the clients of deployed services. The suggested MDECM aims to generate the minimum energy consumption under a particular time constraint and at a minimal cost.

In DECM, for the most part, cloudlets are sent with dynamic programming and conceptualized as a dynamic cloudlet (DCL). Mobile cloud clients convey the service demands through the virtual machine (VM) connected to the client applications through which the demands productively progress to the adjoining cloudlet. The term 'cloudlet' alludes to a layer associating mobile devices with cloud servers in MCC. The suggested MDECM is considering two successful parameters, i.e., service rate and energy to enhance the service performance with limited energy involvement. The parameters employed in the MDECM are as follows:

3.3.1 Service rate

This determines the average quantity of clients that are serviced at a time. The service rate is the volume of service system; should there be a chance that the number of customers to be served at a time is less than the average number of clients arriving, the holding up line will develop infinitely. The service rate relies on the cost and speed of every task. The task service rates are given as the cost ratio (cost essential by the tasks) to the data rate, estimated as follows:

$$S_r = C \diagup L, \tag{10}$$

where S_r is the service rate, C is the cost and L is the speed (data rate).

3.3.2 Energy consumption

The total energy expended for the achievement of a specific requested task can be determined using the formula:

$$E = \frac{k_f - h}{T_E}, \tag{11}$$

where E is the expended energy, k_f is the total task energy, h is the task lost energy, and T_E is the total energy.

The suggested MDECM algorithm is presented in Figure 4. The service rate and energy values are considered for a better service performance with reduced energy and resource allocation cost. The definition of the utilized notations in the suggested MDECM algorithm is presented below.

x Cloudlet node code which refers to the lining up of operations in the cloudlet nodes.

i Choosing which strategy route to be utilized.

M_i Method route which signifies the cloudlet route to be utilized.

t Particular latency or timing cost for every node.

T Execution time unit which refers to the time required to convey services under demands.

S_r Service rate.

E Energy.

t_T Task execution time.

Input: S_r, E, $T_i(x)$, $M_i(x)$ and $N(x)$
Output: *Minimum energy consumptions within a specific timing period and cost reduction.*
1: ***for*** $x \leftarrow 1$ *to* $N(x)$
2: ***for*** $m \leftarrow 1$ *to* $M_i(x)$
3: ***for*** $t_T \leftarrow 1$, *to* $T_i(x)$
4: $S_r \leftarrow C/L$
5: $E \leftarrow \dfrac{k_f - h}{T_E}$
6: ***do*** *the comparison and cancel the pair performing worse*
7: ***end for***
8: ***end for***
9: ***end for***
10: ***for*** $t_T \leftarrow 1$ *to* $Ti(x)$
*11: /*calculate each task service rate and energy consumption from* $N(1)$ *to* $N(x)$
12: ***do*** *the comparison and cancel the pair performing worse */*
13: ***end for***
14: return all results

Figure 4. Modified dynamic energy aware cloudlet model (MDECM) algorithm

The MDECM algorithm comes about lessening the extra energy consumptions amid the wireless communications and continued for the total computation. Using the suggested dynamic programming approach on DCLs plans to select the most proficient communication between mobile devices and cloud servers, and by that resource allocation in a cloud data center, aim to give a high performance without concentrating on assigning VMs to limit energy consumption.

3.3.3 Resource allocation

Resource allocation is the process of doling out and scheduling accessible resources in the most effective and economical manner. The suggested optimized load balancing method accompanied by modified dynamic energy aware cloudlet model solves the issues in resources allocation and results in effective resource management. In MCC, resource management is imperative in sharing computing resources between customer demands. The effective management of accessible resources in the data-center plays a part for both consumer satisfaction and profit maximization. The suggested load balancing method with energy aware model helps cloud administrators in the determination of client's priorities and effective allocation of network resources. The suggested resource allocation procedure is more efficient compared to the existing methods because, in those frameworks, there is no consideration of load balancing over energy reduction amongst the tasks.

4. Results and Discussion

The use of the suggested Krill herd load balancing with viable energy cost aware resource allocation was executed on the JAVA platform. To assess the performance of the suggested model, different parameters such as load balancing, energy consumption, average turn round time, average waiting time, throughput, task energy consumption, execution time, and latency were measured and compared to the prevailing strategies.

4.1 Execution time span

The execution time span is the time taken from the initialization of the first task and the end of the last task. It is determined using the equation:

$$E_t = R_{first} - R_{last}, \tag{12}$$

where E_t is the execution time span, R_{first} is the time of ending the last task, R_{last} is the time of starting the first task. The comparative analysis graph of the execution time of the suggested Krill-LB with the existing Krill herd, HBB-LB Round Robin algorithms is shown in Figure 5.

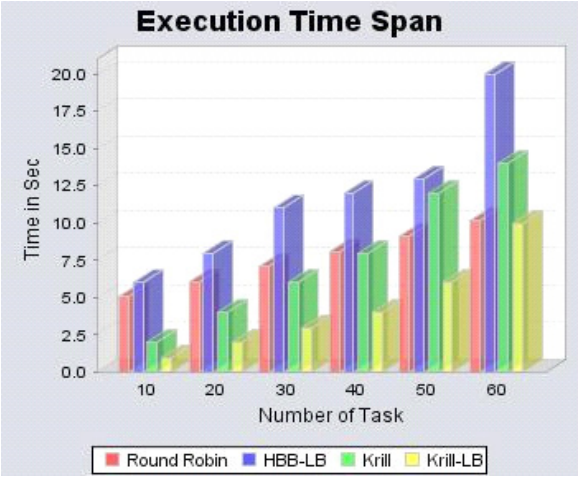

Figure 5. Comparison of the execution time span of the suggested Krill-LB with the existing methods

In Figure 5, the execution time span of the suggested krill-LB was compared to those of Krill herd, HBB-LB, and Round Robin algorithms. The comparison clearly demonstrates that the suggested Krill-LB requires lesser execution time span compared to the existing techniques.

4.2 Energy consumption

The total amount of energy utilized by the methodology in data transfer is termed 'energy consumption'. The performance of the suggested MDECM was compared with the existing DECM which performs analogously to the expected framework. The correlation analysis of the projected MDECM with the DECM is shown in Figure 6.

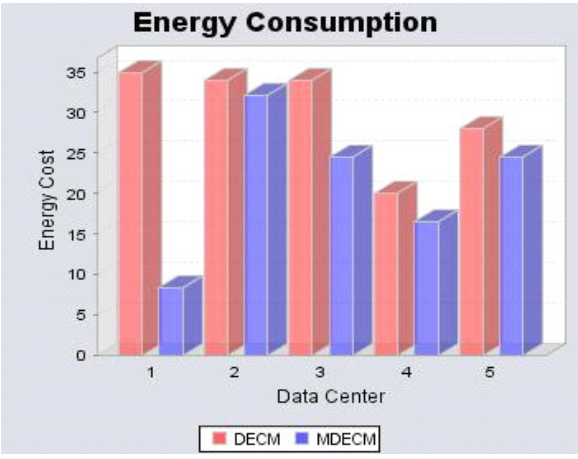

Figure 6: Comparison analysis of the suggested MDECM with DECM for energy consumption

The comparison analysis graph in Figure 6 demonstrates that the projected MDECM requires less energy compared to the DECM for an increasing number of data centers.

4.3 Throughput

The total sum of tasks prepared by specific server is defined as "throughput", which is measured by bit/second. Throughput is an important factor for reliability and system performance assessments. It is determined using the formula:

$$T = \frac{d_t * s}{t_s}, \tag{13}$$

where T is the throughput, d_t is the number of delivered task data, s is the size of data, t_s is the aggregate task simulation time. The comparison of the throughput of the projected Krill-LB, Krill herd, HBB-LB and Round Robin algorithms is presented in Figure 7.

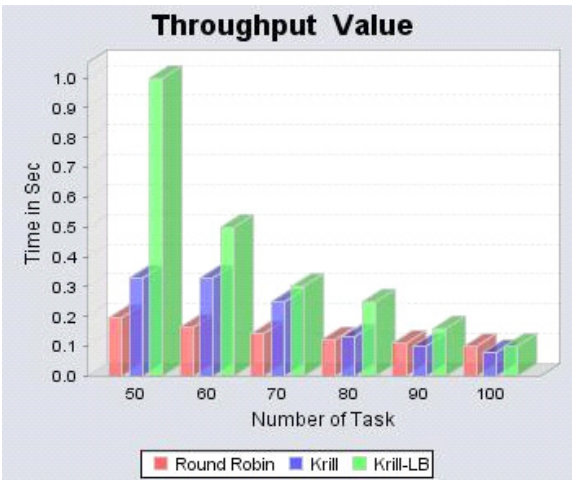

Figure 7. Comparison of the throughput of Krill-LB with the existing algorithms

The comparison graph in Figure 7 demonstrates that the suggested Krill-LB performed better than Krill herd, Round Robin, and Krill herd algorithms in terms of the throughput. The throughput of Krill-LB was higher than those of the benchmarked algorithms.

4.4 Average turn round time

In MCC, turnaround time is the aggregate time between the submission of a task for execution and the arrival of the outcome to the user. The average turnaround time is calculated by the equation:

$$T_{avg} = t_c - t_A, \quad (14)$$

where T_{avg} is the average turnaround time of tasks, t_c is the tasks' completion time, t_A is the tasks' arrival time. The comparison of the suggested Krill-LB with Krill herd, HBB-LB, and Round Robin algorithms in terms of the average turn round time is shown in Figure 8.

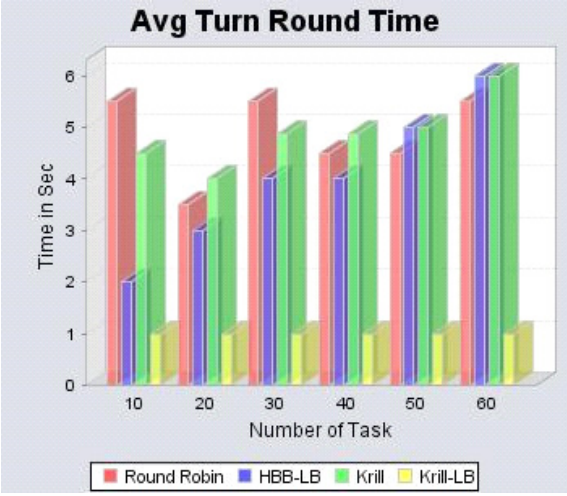

Figure 8. Comparison of Krill-LB with the existing methods in terms of the average turn round time

In Figure 8, the average turnaround time of Krill-LB was compared to the Krill herd, HBB-LB, and Round Robin algorithms. The results showed that the suggested Krill-LB had lesser average turn round time compared to the benchmarked algorithms.

4.5 Average waiting time

The average time frame between a task request and its execution is termed the average waiting time. The average waiting time is calculated using the equation:

$$T_{wait} = T_{avg} - t_{burst}, \quad (15)$$

where T_{wait} is the average waiting time, T_{avg} is the average turnaround time, t_{burst} is the burst time that implies the extent of time the processor utilizes before it is in no way further ready. The comparison of the suggested krill-LB with the existing algorithms in terms of the average waiting time is presented in Figure 9.

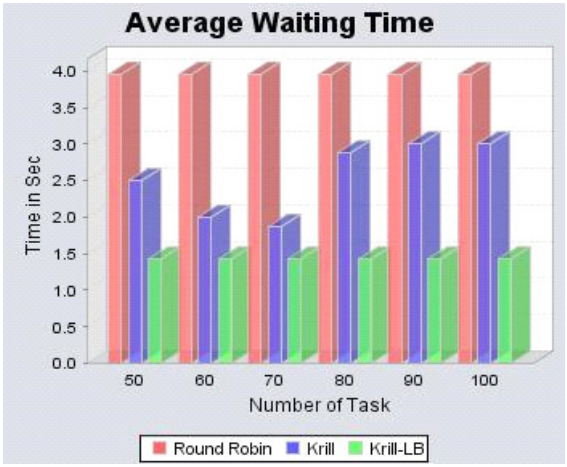

Figure 9. Comparison of the suggested Krill-LB with existing methods in terms of the average waiting time

The comparison graph in Figure 9 demonstrates that the suggested Krill-LB has a lesser waiting time compared to the benchmarked algorithms. Having a lesser waiting time implies the suggested algorithm operates more rapidly than the benchmarked algorithms.

4.6 Task energy consumption

Energy consumption is defined as the quantity of energy utilized by the strategy in data transfer. The task energy consumption stands as the quantity of energy utilized to finish every task provided. The comparison analysis of the suggested MDECM with DECM is presented in Figure 10.

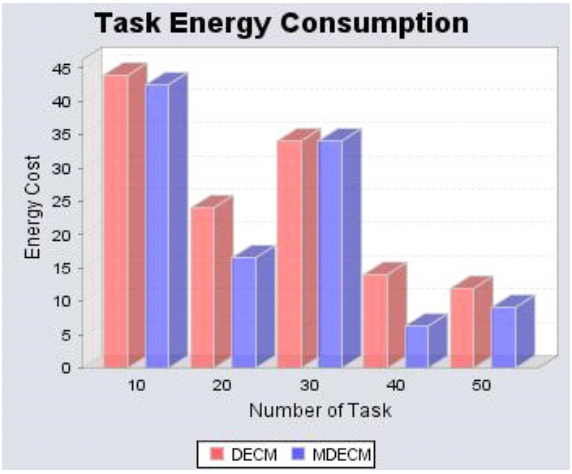

Figure 10. Comparison of the suggested MDECM with DECM for task energy consumption

The comparison graph in Figure 10 demonstrates that the suggested MDECM needs lesser energy consumption for task execution compared to DECM.

4.7 Load balancing

Load balancing is the load in each service measured by using standard deviation. Load balancing is calculated using the formula:

$$L_b = \sqrt{\frac{\sum_{i=1}^{n}(l_i - l)^2}{N}}, \quad (16)$$

where L_b is the load balancing, N is the number of services, l_i is the number of task loading in service i and l is the average load of completed services. The comparison study of the suggested Krill-LB with the existing algorithms is shown in Figure 11.

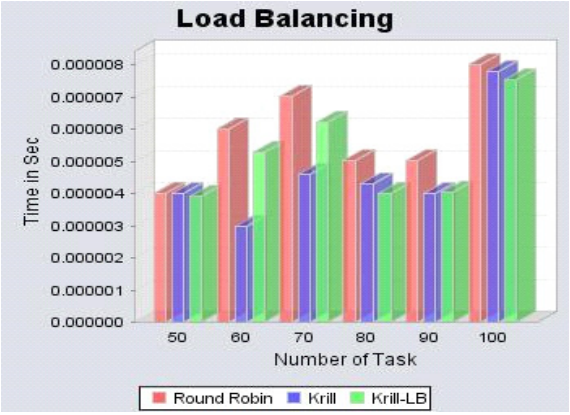

Figure 11. Comparison of the suggested Krill-LB with existing algorithms in terms of load balancing

In Figure 11, the load balancing performance of the suggested Krill-LB was compared to that of some existing algorithms. The results showed that the suggested Krill-LB stands at per with some of the algorithms.

4.8 Latency

Latency is the overall time of finishing a task, accompanied by the delay. This should be low for an effective strategy in the execution process. The latency of tasks is calculated using the equation:

$$L = E_t + D, \quad (17)$$

where L is the latency, E_t is the assessed finishing time of tasks, and D is the delay of task completion. The comparison analysis of the projected Krill-LB with the existing algorithms in load balancing is shown in Figure 12.

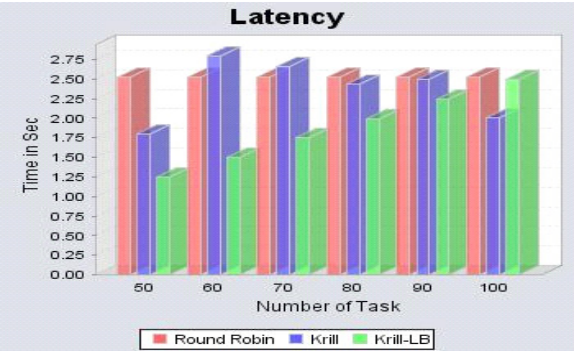

Figure 12. Comparison of the suggested Krill-LB with the existing algorithms in terms of latency

In Figure 12, the latency of the projected Krill-LB was compared to that of some existing algorithms and the results showed the suggested Krill-LB to have a lesser latency compared to benchmarked algorithms.

5. Conclusion

In the suggested energy-cost aware and load balancing procedure, the broker policy has been optimized using the Krill herd optimization algorithm named Krill-LB. The optimized parameter of the broker policy was forwarded to the Modified Dynamic Energy-aware Cloudlet-based Mobile cloud computing model (MDECM) strategy. Finally, the energy cost of the mechanism was calculated for improved load balancing and aimed at the minimal energy requirement for task processing. The use of MDECM can enhance the performance of the method compared to the existing strategies. Additionally, it lessened the computational complication through enhancing the computing ability of the processing features. The use of the suggested algorithm minimized several issues such as energy consumption, average turn round time, average waiting time, execution time, latency, load balancing, task energy and throughput which demonstrates a better performance of the framework.

REFERENCES

1. Banerjee, S. & Hecker, J. P. (2017). A Multi-Agent System Approach to Load-Balancing and Resource Allocation for Distributed Computing. In *First Complex Systems Digital Campus World E-Conference 2015,* Springer.
2. Chen, S.-L., Chen Y.-Y. & Kuo, S.-H. (2016). CLB: A novel load balancing architecture and algorithm for cloud services, *Computers & Electrical Engineering.*
3. Cojoacă, E. Ş. D., Popescu, M. A.-M. & Ambăruş, G. C. (2017). Cloud Computing Technology to Assist Government in Decision Making Process, *Studies in Informatics and Control*, *26*(2), 249-258.
4. Farrag, A. A. S., Mahmoud, S.A. & El Sayed, M. (2015). Intelligent cloud algorithms for load balancing problems: A survey. In *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS).* IEEE.
5. Fernando, N., Loke, S.W. & Rahayu, W. (2013). Mobile cloud computing: A survey, *Future generation computer systems*, *29*(1), 84-106.
6. Gai, K. et al. (2016). Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing, *Journal of Network and Computer Applications*, *59*, 46-54.
7. Gao, J. et al. (2013). Mobile cloud computing research-issues, challenges, and needs. In *2013 IEEE 7th International Symposium on Service-Oriented System Engineering (SOSE)*. IEEE.
8. Gill, Q.K. & Kaur, K. (2016). A computation offloading scheme for performance enhancement of smart mobile devices for mobile cloud computing. In *IEEE International Conference on Next Generation Intelligent Systems (ICNGIS).*
9. Gong, Y. et al. (2015). Protecting location privacy for task allocation in ad hoc mobile cloud computing, *IEEE Transactions on Emerging Topics in Computing*.
10. Gope, P. & Das, A. K. (2017). Robust anonymous mutual authentication scheme for n-times ubiquitous mobile cloud computing services, *IEEE Internet of Things Journal*.
11. Grover, J. & Katiyar, S. (2013). Agent based dynamic load balancing in Cloud Computing. In *2013 International Conference on Human Computer Interactions (ICHCI).* IEEE.
12. Hasan, R. A. K. et al. (2017). A comprehensive study: ACO for Facility Layout Problem. In *2017 16th RoEduNet Conference: Networking in Education and Research*.
13. Kaliappan, M., Augustine, S. & Paramasivan, B. (2016). Enhancing energy efficiency and load balancing in mobile ad hoc network using dynamic genetic algorithms, *Journal of Network and Computer Applications*, *73*, 35-43.
14. Krishna, P. V. (2013). Honey bee behavior inspired load balancing of tasks in cloud computing environments, *Applied Soft Computing*, *13*(5), 2292-2303.
15. Li, J. et al. (2017). Computation Partitioning for Mobile Cloud Computing in a Big Data Environment, *IEEE Transactions on Industrial Informatics*, *13*(4), 2009-2018.
16. Li, K. et al. (2011). Cloud task scheduling based on load balancing ant colony optimization. In *2011 Sixth Annual Chinagrid Conference (ChinaGrid)*. IEEE.
17. Liang, H. et al. (2012). An SMDP-based service model for interdomain resource allocation in mobile cloud networks, *IEEE transactions on vehicular technology*, *61*(5), 2222-2232.
18. Liu, Y. & Lee, M. J. (2015). An adaptive resource allocation algorithm for partitioned services in mobile cloud computing. In *2015 IEEE Symposium on Service-Oriented System Engineering (SOSE)*. IEEE.
19. Liu, Y., Lee, M. J. & Zheng Y. (2016). Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system, *IEEE Transactions on Mobile Computing*, *15*(10), 2398-2410.
20. Malekloo, M. (2015). *Multi-objective ACO resource consolidation in cloud computing environment*. École de technologie supérieure.
21. Merezeanu, D., Vasilescu, G. & Dobrescu, R. (2016). Context-aware Control Platform for Sensor Network Integration in IoT and Cloud, *Studies in Informatics and Control*, *25*(4), 489-498.
22. Milani, A. S. & Navimipour, N. J. (2016). Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends, *Journal of Network and Computer Applications*.

23. Mohammed, M. A. & Hasan, R. A. K. (2017). Particle Swarm Optimization for facility layout problems FLP – A comprehensive study. In *2017 IEEE 13th International Conference on Intelligent Computer Communication and Processing.*

24. Mohammed, M. N. & Hammood, O. A. (2017). Hybrid LTE-VANETs Based Optimal Radio Access Selection. In *Proceedings of the 2nd International Conference of Reliable Information and Communication Technology (IRICT), Recent Trends in Information and Communication Technology*. Springer.

25. Naha, R. K. & Othman M. (2016). Cost-aware service brokering and performance sentient load balancing algorithms in the cloud, *Journal of Network and Computer Applications*, *75*, 47-57.

26. Rashidi, S. & Sharifian, S. (2017). A hybrid heuristic queue based algorithm for task assignment in mobile cloud, *Future Generation Computer Systems*, *68*, 331-345.

27. Sarrab, M. & Bourdoucen, H. (2015). Mobile Cloud Computing: Security Issues and Considerations, *Journal of Advances in Information Technology, 6*(4).

28. Sharkh, M. A. & Shami, A. (2017). An evergreen cloud: Optimizing energy efficiency in heterogeneous cloud computing architectures, *Vehicular Communications.*

29. Wang, K., Yang, K. & Magurawalage C. (2016). Joint energy minimization and resource allocation in C-RAN with mobile cloud, *IEEE Transactions on Cloud Computing*.

30. You, C. et al. (2017). Energy-efficient resource allocation for mobile-edge computation offloading, *IEEE Transactions on Wireless Communications*, *16*(3), 1397-1411.