Rev. Roum. Sci. Techn.– Électrotechn. et Énerg.
Vol. **69**, *1*, pp. 79–84, Bucarest, 2024

LOAD BALANCING IN CLOUD COMPUTING VIA MAYFLY OPTIMIZATION ALGORITHM

MARIA JESI [1], AHILAN APPATHURAI [2], MUTHUKUMARAN NARAYANAPERUMAL [3], ARUL KUMAR [4]

Keywords: Cloud computing; Load balancing; Mayfly optimization; Task scheduling.

Cloud computing is a new technology that enables users to store and retrieve data via the Internet on demand rather than using their hardware. Cloud computing comprises distinct data centers (servers) and clients (users). Load unbalancing is a multi-variant, multi-constraint issue that lowers the efficacy and performance of system resources. Therefore, a load scheduling technique is needed to distribute work among the right VMs and preserve the trade-off between them. To achieve better performance, this paper presents a novel mayfly optimization algorithm for load balancing (MFO-LB), which utilizes mayfly flight behavior and mating dynamics. The proposed technique balances the load in the cloud by managing the incoming loads by allocating resources according to user requests. The proposed work intends to increase performance by uniformly dividing the workload among the virtual machines, which will decrease utilization and reaction time. The proposed MFO-LB approach is beneficial for maintaining system stability, reducing response time (RT), and maximizing resource productivity in cloud environments. Finally, the effectiveness of the proposed technique is assessed by employing several metrics, including execution cost, RT, execution time, and makespan. The proposed method achieves up to 23.4 % low RT, a 24 %decrease in makespan, and a 31.5 % decrease in completion time, respectively.

1. INTRODUCTION

Cloud computing (CC) is an internet-based technology that has seen rapid growth in communication breakthroughs by making computing resources and services available online to clients with various needs [1]. The cloud computing model encompasses the hardware and software data centers use to distribute applications over the Internet [2]. In general, users anticipate a particular quality of service (QOS), which can be achieved by utilizing an emerging technology known as virtualization [3]. A virtual machine (VM) is an implementation of cloud computing that provides compute resources in virtual resources [4–6]. Cloud service providers store data on several cloud servers or data centers (DCs). Tasks are allocated to distinct servers using virtual machines in response to requests from cloud providers [7]. Virtual machines are assigned different tasks. Distributed computing supports virtual machines (VMs), and all incoming work is distributed among them. Some VMs may be overworked, while others may need to be more utilized when duties are distributed among VMs.

A key role of cloud computing is to optimize performance and productivity by efficiently sharing resources [8]. The cloud enables cloud providers to set up contemporary data centers and users to get their fields working on the cloud at an affordable cost [9,10]. Cloud providers provide vital services to users who face challenging tasks due to limited resources. The cloud service provider's (CSP) role in providing services to users is exceedingly complicated with the accessible virtual cloud resources [11]. The growing number of cloud clients has resulted in a significant demand for computer resources. The phrase "load" can refer to memory, storage, or CPU, as well as network load [12].

Load balancing (LB) spreads workloads to ensure that every data center is idle, under-loaded, or overloaded [13]. LB in a cloud domain aims to ensure that no virtual machines are overloaded while others are overloaded or doing nothing [14,15]. In LB, the processing time for applications is reduced. LB can effectively reduce energy consumption and provide QoS because it distributes the load and decreases resource consumption [16]. The cloud service provider (CSP) and the cloud service user benefit from LB [17].

While processing user requests, some VMs receive a high volume of tasks, while others receive fewer tasks. As a result, CSP has unbalanced machines with a large gradient in user tasks and resource consumption [18,19]. So, in a cloud environment, we need a capable machine to schedule the work [30,31]. Various factors, including turnaround time and RT, must be considered when developing an LB algorithm. Many optimization techniques have been employed for LB in cloud environments, such as particle swarm optimization (PSO) [29], firefly optimization (FFO) [28], ant colony optimization (ACO), *etc*., but they lack accuracy in allocating resources effectively. This paper proposes a novel mayfly optimization technique for load balancing (MFO-LB) to solve these problems. The major goal of the proposed work is given as follows,

- The proposed MFO-LB method creates an objective function based on three variables, execution time, execution cost, and load, to assign jobs to VMs based on capacity.
- MFO-LB is based on the mayfly flight behavior and mating process to manage the incoming load.
- The proposed work intends to increase performance by uniformly dividing the workload among the virtual machines, reducing utilization and reaction time.
- The performance of the proposed method is evaluated using specific parameters such as latency, execution time, RT, and cost.

The remaining section of the paper has been arranged as follows: The related work is briefly presented in section 2. Section 3 presents the suggested methodology. The detailed presentation of the experiment's findings in section 4 demonstrates the effectiveness of the suggested strategy. Section 5 brings the suggested methodology to a close.

[1] Department of Computer Science Engineering, Loyola Institute of Technology and Science, Thovalai, Nagercoil, Tamil Nadu, India (Corresponding author)

[2] Department of Electronics and Communication Engineering, PSN College of Engineering and Technology, Tirunelveli, Tamil Nadu, India

[3] Centre for Computational Imaging and Machine Vision, Department of Electronics and Communication Engineering, Sri Eshwar College of Engineering, Coimbatore, Tamil Nadu, India

[4] Department of Information Technology, Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam, Chennai, Tamil Nadu, India
E-mails: mariajesi.cse@lites.edu.in, akhilanappathurai@psncet.ac.in, muthukumaran.n@sece.ac.in, arulkumarv@ssn.edu.in

DOI: 10.59277/RRST-EE.2024.69.1.14

2. LITERATURE SURVEY

LB is one of the major issues in cloud computing. Numerous experts have devised strategies to preserve LB across virtual servers. Some of these are covered in this section.

Reference [7] introduced a chaos social spider algorithm (CSSA) to schedule user workloads in cloud VMs with balanced work distribution. The proposed solution would shorten the cloud scheduling procedure, enhancing the cloud system's throughput. Experimental results show that the proposed CSSA reduces make time by 14.8 percent compared to ABC, PSO, GA, and HFKCS for 100–1000 tasks.

Reference [20] presented an LB method based on constraint measure (CMLB). According to the experiment results, the suggested CMLB performs better by migrating only three jobs, while the existing LB technique, which includes HDLB, DLB, and HBB-LB, moves many tasks.

Reference [18] introduced a modified honeybee-inspired method for better allocating LB and resources. By dividing tasks over many networks, this strategy ensures that resources are not underused or abused. It also shortens the virtual machine's reaction time and evaluates each cloudlet's workload.

Reference [17] introduced a binary variation of the bird swarm optimization LB (BSO-LB) approach based on three bird-mimicking behaviors. Using an appropriate fitness function can reduce the makespan and increase resource utilization.

Reference [10] presented a dynamic scheduling system that determines the compatibility of requests and virtual machines. The proposed technique might reduce RT and makespan by evenly distributing requests among the VMs.

Reference [6] proposed a hybrid LB method using pigeon-inspired optimization (PIO) and Harries Hawks optimization (HHO) to improve RTs for customers when they request something. Compared to existing techniques, the proposed method significantly balances the virtual machine load in less time. The suggested method is 97 % more efficient than the state-of-art method.

Reference [22] proposed an updated LB algorithm (LBA) considering QoS task parameters, virtual machine priority, and resource allocation. The LB algorithm suggests 78 % more resources than the dynamic LBA algorithm.

Reference [27] suggested LB as a metaheuristic optimization strategy to mitigate scheduling difficulties and enhance cloud infrastructure service provider performance. The suggested method is divided into two stages: statically constrained metaheuristic optimization (MHOS-S) and dynamically constrained metaheuristic optimization (MHO-D), which deal with the problem's dynamic aspects and consider its static properties. According to the results, the suggested technique performs better in difficult settings than current metaheuristic algorithms.

Reference [15] developed multi-objective scheduling using particle swarm optimization (MOSPSO) for activity development to provide optimal resource allocation. The preliminary results suggest that MOSPSO's calculation may reduce makespan by up to 50 %.

Reference [5] presented spider monkey optimization-inspired LB (SMO-LB). The experimental results demonstrate that this technique decreases task average RT by 10.7 s and makespan by 21.5 s compared with existing methods.

From the above-reviewed methods, the researchers focus on minimizing the RT and makespan time in the cloud. This is still a difficult task. Our proposed MFO-LB method overcomes these challenges by using a mayfly optimization algorithm. The proposed framework is presented in the following section.

3. PROPOSED LOAD BALANCING TECHNIQUE

The mayfly optimization method for load balancing technology (MFO-LB), which is based on the flying and mating behaviors of mayflies, is described in this section. The suggested method combines the concepts of evolutionary and swarm intelligence algorithms. The purpose of this process is to distribute the task to a VM using the mayfly optimization algorithm (MFO) to reduce execution times and costs while balancing the load. Using the suggested LB technique, all VMs in the cloud are monitored in real-time. The suggested load-balancing architecture is depicted as a system model in Fig. 1.

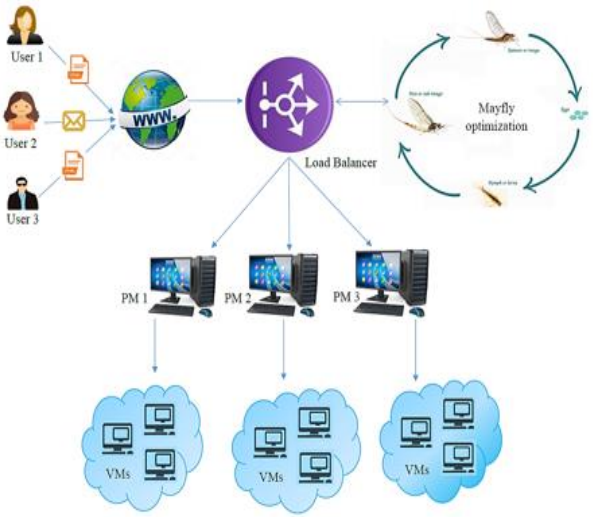

Fig. 1 – MFO-LB architecture.

3.1. PROBLEM STATEMENT

Using cloud D as an example, which consists of x physical machines or any actual machine made up of X VMs, any physical machine is also made up of many virtual machines in the same way: The first virtual machine is denoted by VM1, and the final by VMx (VM). Similarly, if users make up the cloud, the user function can be described as:

$$\boldsymbol{S_i} = \{\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots \boldsymbol{t_z}\}. \quad (1)$$

The major objective of this process is to maintain an optimal load across all VMs in a cloud environment, limit expenses and energy use, and maximize resource utilization and QoS. This research has suggested an effective multi-purpose approach based on the mayfly optimization algorithm to address this issue.

3.2. MAYFLY OPTIMIZATION ALGORITHM FOR LOAD BALANCING

The LB has been the problem of the mating of mayflies. The users' tasks should be to find the correct sources of VMs. To implement MFO in LB, the cloud environment should be reconfigured based on matching assumptions. The processing ability of virtual machines is based on their allocated resources (RAM, processing units, and bandwidth) relative to the projected traffic flow length. Each pair represents a user task and a virtual machine (VM). A load

balancer can achieve an optimal balancing state using this mating process, as shown in Fig. 2.

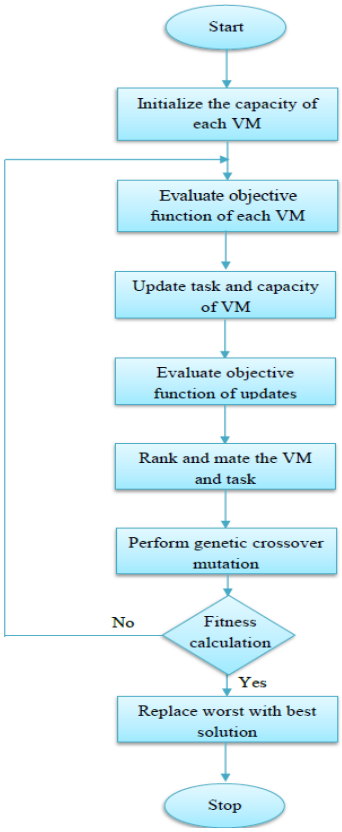

Fig. 2 – Flow diagram of mayfly optimization.

The mayfly optimization technique begins with the social behavior of mayflies. As each job in the VM is assigned, the VM constantly integrates tasks based on its balancing state (overloaded, balanced, or underloaded). Males and females initially received distinct virtual machines (VM) and user tasks. After evaluating the fitness function, update the capacity of the virtual machine. Once the VM's capacity has been updated, rank and match the tasks and VM, then determine the fitness function. Once the best solution has been identified, effectively replace the worst option with the best one while allocating many resources.

Step 1: Initialization

In this step, the male and female populations are randomly assigned. There are many solutions in the population. User tasks and virtual machines (VM) were used to build the solution. VMs are initially randomly assigned tasks.

Step 1.1: Fitness calculation

A fitness function is then used to update the solutions. As a result, each mayfly is randomly placed as a candidate solution in the problem space, represented as a dim-dimensional vector,

$$\mathbf{F} = (F_1, F_2, \ldots \ldots F_{dim}). \quad (2)$$

A may fly's velocity can be described as the alteration in direction and position of every flight during flight, as well as a dynamic interaction of both social flying and individual experience as

$$\boldsymbol{V} = (V_1, V_2, \ldots \ldots V_{dim}). \quad (3)$$

Step 2: Update of the velocity of Male Flies

Male mayflies are always effective and migrate on their own choice. If $z_a(t)$ is the actual situation of the male mayfly and j denotes the searching space at iteration t, the position of the mayfly is altered by adding a velocity $g_a(t+1)$ to the current position from

$$z_a(t+1) = z_a(t+1) + g_a(t+1). \quad (4)$$

The mayflies' movement on the top of the water to dance is arithmetically modelled where $g_a(t+1)$ is given according to the following equation:

$$g_{ab}(t+1) = g_{ab}(t) + m_1 e^{-\beta s_p^2}\left(p\text{best}_{ab} - z_{ab}(t)\right) + \\ + m_2 e^{-\beta s_f^2}\left(f\text{best}_a - z_{ab}(t)\right). \quad (5)$$

A positive constant m_1 and m_2 are utilized to scale the social and cognitive contributions respectively. In addition, p_{best} is the best mayfly position that a have ever visited. Mayflies are hidden from others by using the visibility coefficient β. In the dimension b, $z_{ab}(t)$ and $g_{ab}(t)$ represent the candidate's velocity and position.

Step 2.1: Evaluate the fitness of male flies

The Cartesian distance between z_a with $p\text{best}_a$ and xi z_a with f_{best} is described by the variables s_p and s_f. These distances are determined using the formula:

$$|Z_a - z_a| = \sqrt{\sum_{b=1}^{n}(Z_{ab} - z_{ab})^2}. \quad (6)$$

The best position $p\text{best}_{ab}$ at time t+1 is estimated as follows in the case of minimization problems:

$$p\text{best}_a = \begin{cases} z_a(t+1), & \text{if } l(z_a(t+1)) < l(p\text{best}_a) \\ \text{kept same}, & \text{otherwise} \end{cases}. \quad (7)$$

The global best position $f\text{best}_a$ at time t, is represented as follows:

$$f\text{best}_a \in \{p\text{best}_1, p\text{best}_2, \ldots p\text{best}_n / l(p\text{best}_.))\} = \\ \min\{l(p\text{best}_1), l(p\text{best}_2), \ldots l(p\text{best}_n)\}. \quad (8)$$

The algorithm must function properly so that the finest mayflies achieve their up-and-down nuptial dance. Consequently, the most effective mayflies must constantly alter their velocities, which are calculated as:

$$g_{ab}(t+1) = g_{ab}(t) + nc \cdot s. \quad (9)$$

Here *nc* represents the coefficient of nuptial dance, and *s* denotes the random number between [–1, 1]. By moving up and down in this way, the algorithm takes on a stochastic component.

Step 3: Updating of the velocity of Female Mayflies

Females are not grouped like males. Males fly around females to breed. By adding a velocity $u_a(t+1)$ to the position s that has been updated via the following equation, with $u_a(t)$ representing the a-th female candidate path in the solution space:

$$u_a(t+1) = u_a(t+1) + g_a(t+1). \quad (10)$$

Males usually breed with females in order of quality, then the second-best male with the second-best female and so on. As a result, the velocity was determined as follows:

$$g_{ab}(t+1) = \\ = \begin{cases} g_{ab}(t) + m_2 e^{-\beta s_{ql}^2}\left(g_{ab}(t) - u_{ab}(t)\right), & \text{if } l(u_a) > l(z_a) \\ g_a(t) + lq \cdot s, & \text{if } l(u_a) \le l(z_a) \end{cases}. \quad (11)$$

where $g_{ab}(t)$ is the female mayfly velocity a in length b = 1, 2,...N at time t, the female mayfly position a in the length b at time t is given by $u_{ab}(t)$, $m_2$is the optimistic attractor constant and the coefficient of fixed visibility is defined as β, and s_{ql} is defined as the female and male mayflies cartesian distance, thus it is estimated as eq. (11). The lq parameter determines when a female is not attracted to a male and so flies at random, while s is a random number between [–1, 1].

Step 4: Genetic and crossover

Once the velocity and position of the individuals have been updated, the topmost half will be renamed as male mayflies, and the other half will be renamed female mayflies. According to the mayfly mating procedure, two mayflies must mate in the following order: Females with the best relationships date the best men, females with the second-best relationships date the second-best men, and so on. As a result of these crossings, two offspring are formed using:

$$offspring1 = I \cdot \text{male} + (1 - I) \cdot \text{female} \quad (12)$$

$$offspring2 = I \cdot \text{female} + (1 - I) \cdot \text{male}. \quad (13)$$

Individuals and offspring of half-named male and female mayflies will be sorted with their alternates. The best candidates can reach the global optima and thus identify the solution after multiple iterations. Mayflies were chosen as the most effective optimization method because they combine the advantages of swarm intelligence and evolutionary algorithms, which achieves a better balance of exploitation and exploration. Therefore, by mimicking this behaviour in the cloud, LB will become more dynamic.

4. RESULT AND DISCUSSION

This section examines the proposed MFO-LB technique using several factors like makespan, execution time, RT, completion time, and resource utilization by several tasks. It's implemented in Java and tested with Cloudsim on several different cloud settings. The experiment setup and algorithm validation will be established initially in this simulation algorithm. The Cloudsim simulator is used for this simulation setup, and each VM has the same configuration, requiring the cloud service provider to establish the threshold value and the VM capabilities. In this study, we develop a generic scenario in which ten virtual machines are produced and one data center is constructed. These virtual machines (VMs) are to be installed on every host that the data center creates using algorithms; each host is assigned a PES (processing core) variation number.

4.1. PERFORMANCE METRICS

The effectiveness of the developed strategy has been assessed using specific metrics such as makespan, RT, execution time, completion time, latency, VM migration, and computational cost to demonstrate the efficiency. The performance metrics are explained as follows.

4.1.1. Makespan

The time it takes a virtual machine (VM) to complete all tasks in the task queue is known as its makespan. When a job has certain goals to achieve, the cloud service provider is responsible for determining the systems' make span in numerous scenarios [21]. A minimum makespan implies efficient mapping of user tasks to CNs. Makespan is computed based on

$$\text{Makespan} = \text{Completion time of last server} - \\ -\text{starting time of first server}. \quad (14)$$

4.1.2. Response time

It is the time taken to balance the load and respond to a user's query by allocating the load VMs with the minimum number of tasks. The correlation between a system's reaction time and efficacy is inverse. The optimal *RT* is equal to the optimum makespan value.

$$\text{RT} = \text{Execution time} + \text{waiting time} \quad (15)$$

4.1.3. Execution time

It measures how long a virtual machine (VM) takes to run or complete a task. The average execution time is calculated by applying:

$$\text{ET} = \frac{T_a}{L_b}. \quad (16)$$

4.1.4. Completion time

The addition of the execution time, waiting time, and transmission time of the X[th] task on Y[th] virtual machine provides the completion time of a task allocated to a VM which is stated as:

$$\text{CT} = \text{Execution time} + \text{Transmission time} + \\ +\text{Waiting time}. \quad (17)$$

4.1.5. Latency

The delay in the LB system for the number of requests (N), Processing time per request (P) network latency (N_L), and Queueing Time (Q) is referred to as latency (L). A system with higher efficiency has lower latency,

$$L = N \cdot P + N_L + Q. \quad (18)$$

4.1.6. VM migration

Migration (M) is the process of transferring a virtual machine from one actual hardware environment to another when the number of servers (S) is full. An unbalanced system frequently migrates virtual machines.

$$M = S \cdot VM \cdot \left(\frac{\text{Number of servers exceeding thresholds}}{\text{Total number of servers}}\right). \quad (19)$$

4.1.7. Computational cost

It is the simulation of the execution time during the step in which the target process is time-measured. There may be a charge if certain tasks are to be accomplished.

$$\text{Total Cost} = \text{Execution time} + \\ +\text{Charge associated with task}. \quad (20)$$

4.2. COMPARATIVE ANALYSIS

The efficiency of the suggested strategy has been demonstrated by evaluating its efficacy against established methodologies [25,26]. For comparative analysis, four state-of-the-art algorithms, including the MOSPSO, MHO, SMO-LB, and BSO-LB are chosen in this study.

As shown in Fig. 3, the suggested technique has been compared with the state-of-arts methods. According to Figure 3, the MFO-LB technique allows VMs to be placed optimally in a greater number of tasks than existing systems. When compared to BSO-LB, the proposed model reduces the makespan by 25 %. Figure 4 shows how the proposed method compares to other current systems in terms of RT. In comparison with existing techniques, the MFO-LB approach is faster to converge and is more accurate at assigning tasks to VMs.

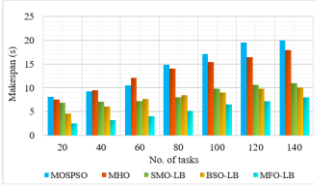

Fig. 3 – Makespan comparison.

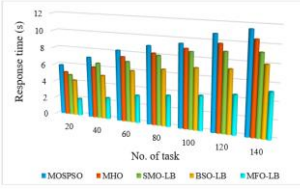

Fig. 4 – Comparison of response time.

This figure compares the suggested strategy to alternative methods in terms of execution time Fig. 5. By optimizing VM placement scheduling, the proposed MFO-LB technique can support both static and dynamic tasks and thus provides faster execution times than other state-of-the-

art approaches. As a result, MFO-LB reduces execution times and schedules tasks faster than any other method.

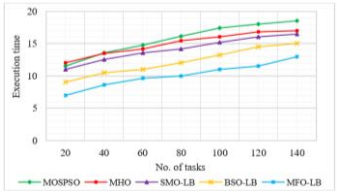

Fig. 5 – Execution time comparison with existing techniques.

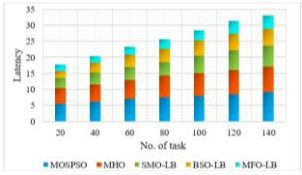

Fig. 6 – Latency comparison.

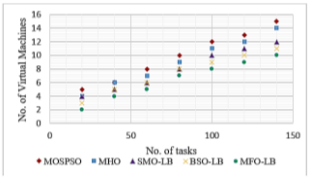

Fig. 7 – No. of task vs virtual machines.

From Fig. 6, the latency for 140 users for the proposed technique is about 4.2 s, 5.2 s for BSO-LB. The comparative study demonstrates that the suggested technique has a lower latency than the others, thus illustrating its usefulness. Following the analysis, the technique presented secures the load-balancing system from latency. The proposed MFO-LB approaches are compared to existing systems based on the number of VMs they utilize. The comparison is significant considering that service providers hoped to deliver services with fewer virtual machines to handle more customer tasks. As shown in Fig. 7, the MFO-LB approach optimally schedules jobs. Therefore, the suggested strategy uses fewer virtual machines than traditional methods. According to Figure 8, the resource usage graph of the suggested system is compared with the existing system, which indicates how efficiently resources are being used. Resource utilization is maximized to 70.01 percent using the proposed approach. Iteration is shown on the x-axis, and resource use is shown on the y-axis. Figure 9 compares the number of VM migrations accomplished with the suggested approach and other approaches. Using the proposed MFO-LB method, virtual machines are balanced, resulting in fewer VM migrations and less PM overload. The proposed method produces a small number of VM migrations.

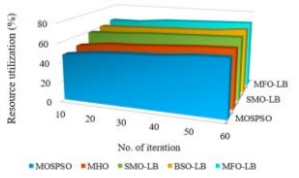

Fig. 8 – Comparison of resource utilization.

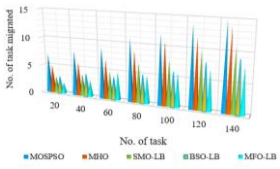

Fig. 9 – Task migrated comparison.

Figure 10 shows the computational complexity and computational cost of the suggested MFO-LB method and the existing algorithm. The data can be complicated and heterogeneous, such as more efficient information [20]. The distributed control method will be converted by using nodes with high-complexity devices. Figure 10 shows that the computational complexity obtained by MOSPSO, MHO, SMO-LB, BSO-LB, and MFO-LB is 59 %, 50 %, 42 %, 32 % and 17 % respectively. the computational complexity obtained by MOSPSO, MHO, SMO-LB, BSO-LB, and MFO-LB is 61 %, 54 %, 40 %, 35 and 20% respectively. It can be observed that MFO-LB outperforms the other methods.

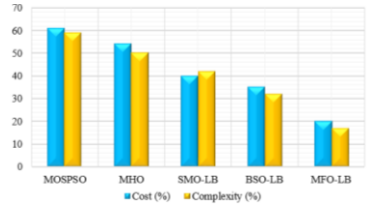

Fig. 10 – Computational complexity and computational cost analysis.

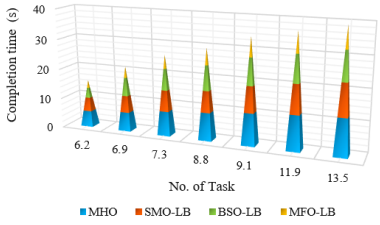

Fig. 11 – Comparison of completion time.

In Fig. 11, the MOSPSO, MHO, SMO-LB, BSO-LB, and MFO-LB methods are compared in terms of completion time. The amount of time needed to perform a task increases along with its number. Among the recognized techniques, the MOSPSO technique has the fastest completion time. The physical machine within the data center is experiencing an uneven workload. As a result of effective LB and task allocation mechanisms, the MFO-LB takes 31 % less completion time than the BSO-LB. Therefore, the suggested method greatly decreases the task completion time.

5. CONCLUSION

This paper proposed a mayfly optimization algorithm for load balancing (MFO-LB), which is based on the behavior of the mayfly and the mating process. A mathematical model for task mapping is employed by the MFO-LB algorithm in a cloud environment and has also been developed to make use of mayfly Optimization to determine the ideal load balance between virtual machines. The findings demonstrate that, in terms of RT, makespan, and completion time, the recommended work outperforms the current tactics. The suggested approach reduces reaction time by up to 23.4 %, makespan by 24 %, and completion time by up to 31.5 %, in that order. Future work will include incorporating intelligent optimization methods for efficient LB to decrease the imbalance, and the results will be applied in real-world applications for throughput, makespan, and RT.

ACKNOWLEDGEMENTS

The Author with a deep sense of gratitude would thank the supervisor for his guidance and constant support rendered during this research.

Received on 3 February 2023

REFERENCES

1. S. Afzal, G. Kavitha, *Optimization of task migration cost in infrastructure cloud computing using IMDLB algorithm*, 2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET), pp. 1–6 (2018).
2. S. Afzal, G. Kavitha, *Load balancing in cloud computing–A hierarchical taxonomical classification*, Journal of Cloud Computing, **8**, *1*, pp. 1–24 (2019).
3. A. Ahilan, P. Deepa, *A reconfigurable virtual architecture for memory scrubbers (VAMS) for SRAM based FPGA's*, International Journal of Applied and Engineering Research, **10**, *10*, pp. 9643–9648 (2015).

4. M. Alouane, H. El Bakkali, *Virtualization in cloud computing: NoHype vs HyperWall new approach*, 2016 International Conference on Electrical and Information Technologies (ICEIT), pp. 49–54 (2016).
5. S. Alshattnawi, M. Al-Marie, *Spider monkey optimization algorithm for load balancing in cloud computing environments*, International Arab Journal of Information Technologies, **18**, *5*, pp. 730–738 (2021).
6. G.A.P. Princess, A.S. Radhamani, *A hybrid meta-heuristic for optimal load balancing in cloud computing*, Journal of Grid Computing, **19**, *2*, pp. 1–22 (2021).
7. V.M.A. Xavier, S. Annadurai, *Chaotic social spider algorithm for load balance aware task scheduling in cloud computing*, Cluster Computing, **22**, *1*, pp. 287–297 (2019).
8. K. Balaji, *Load balancing in cloud computing: issues and challenges*, Turkish Journal of Computer and Mathematics Education (TURCOMAT), **12**, *2*, pp. 3077–3084 (2021).
9. A. Chawla, N. S. Ghumman, *Package-based approach for load balancing in cloud computing*, Big Data Analytics, pp. 71–77 (2019).
10. F. Ebadifard, S.M. Babamir, *Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment*, Cluster Computing, **24**, *2*, pp. 1075–1101 (2021).
11. M. Gamal, R. Rizk, H. Mahdi, B.E. Elnaghi, *Osmotic bio-inspired load balancing algorithm in cloud computing*, IEEE Access, **7**, pp. 42735–42744 (2019).
12. N.J. Navimipour, F.S. Milani, *A comprehensive study of the resource discovery techniques in peer-to-peer networks*, Peer-to-Peer Networking and Applications, **8**, *3*, pp. 474–492 (205).
13. B. Jana, M. Chakraborty, T. Mandal, *A task scheduling technique based on particle swarm optimization algorithm in cloud environment*, Soft Computing: Theories and Applications, Springer, Singapore, 2019, pp. 525–536.
14. A. Kaur, B. Kaur, P. Singh, M.S. Devgan, H.K. Toor, *Load balancing optimization based on deep learning approach in cloud environment*, International Journal of Information Technology and Computer Science, **12**, *3*, pp. 8–18 (2020).
15. N. Malarvizhi, J. Aswini, S. Sasikala, M.H. Chakravarthy, E.A. Neeba, *Multi-parameter optimization for load balancing with effective task scheduling and resource sharing*, Journal of Ambient Intelligence and Humanized Computing, 2021, pp. 1–9.
16. S.T. Milan, L. Rajabion, H. Ranjbar, N.J. Navimipour, *Nature inspired meta-heuristic algorithms for solving the load-balancing problem in cloud environments*, Computers & Operations Research, **110**, pp. 159–187 (2019).
17. K. Mishra, S.K. Majhi, *A binary bird swarm optimization-based load balancing algorithm for cloud computing environment*, Open Computer Science, **11**, *1*, pp. 146–160 (2021).
18. S.K. Mishra, B. Sahoo, P.P. Parida, *Load balancing in cloud computing: a big picture*, Journal of King Saud University – Computer and Information Sciences, **32**, *2*, pp. 149–158 (2020).
19. G. Muthusamy, S.R. Chandran, *Cluster-based task scheduling using K-means clustering for load balancing in cloud datacenters*, Journal of Internet Technology, **22**, *1*, pp. 121–130 (2021).
20. V. Polepally, K.S. Chatrapati, *Dragonfly optimization and constraint measure-based load balancing in cloud computing*, Cluster Computing, **22**, *1*, pp. 1099–1111 (2019).
21. H. Ren, Y. Lan, C. Yin, *The load balancing algorithm in cloud computing environment*, Proceedings of 2012 2nd International Conference on Computer Science and Network Technology, pp. 925–928 (2012).
22. D.A. Shafiq, N.Z. Jhanjhi, A. Abdullah, M.A. Alzain, *A load balancing algorithm for the data centres to optimize cloud computing applications*, IEEE Access, **9**, pp. 41731–41744 (2021).
23. A.K. Sharma, K. Upreti, B. Vargis, *Experimental performance analysis of load balancing of tasks using honeybee inspired algorithm for resource allocation in cloud environment*, Materials Today: Proceedings, 2020.
24. S.G. Sophia, K.K. Thanammal, *An improved homomorphic encryption technology for the surveillance of cloud data*, Solid State Technology, **63**, *2s*, pp. 2671–2674 (2020).
25. A. Ullah, *Artificial bee colony algorithm used for load balancing in cloud computing*, IAES International Journal of Artificial Intelligence, **8**, *2*, p. 156 (2019).
26. Y. Zhu, P. Liu, *Multi-dimensional constrained cloud computing task scheduling mechanism based on genetic algorithm*, International Journal of Online Engineering, **9**, pp. 15–18 (2013).
27. S. Ziyath, S. Senthilkumar, *MHO: meta heuristic optimization applied task scheduling with load balancing technique for cloud infrastructure services*, Journal of Ambient Intelligence and Humanized Computing, **12**, *6*, pp. 6629–6638 (2021).
28. N. Dif, E. Boudissa, M. Bounekhla, I. Dif, *Firefly algorithm improvement with application to induction machine parameters identification*, Rev. Roum. Sci. Techn. – Électrotechn. et Énerg., **65, *1*,** pp.35–40 (2020).
29. A. Zangeneh, *Optimal design of onshore wind farm collector system using particle swarm optimization and Prim's algorithm*, Rev. Roum. Sci. Techn. – Électrotechn. et Énerg.,**64, *4***, pp. 349–356 (2019).
30. M. Balakrishnan, M. Nalina, K. Ramya, K. Senthilsriram, *Cloud computing+ based data validation and migration in ETL using talend.* In 2022 6th International Conference on Electronics, Communication and Aerospace Technology, IEEE, pp. 1349–1355 (2022).
31. M.S. Kumar, F.D. Shadrach, S.R. Polamuri, R. Poonkodi, V.N. Pudi, *A binary bird swarm optimization technique for cloud computing task scheduling and load balancing*. 2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES). IEEE, pp. 1–6, 2022.
32. R. Raja Kumar, J. Athimoolam, A. Appathurai, S. Rajendiran, *Novel segmentation and classification algorithm for detection of tomato leaf disease.* Concurrency and Computation: Practice and Experience, **35,** *12*, 7674 (2023).