International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 / https://doi.org/10.14445/22315381/IJETT-V70I6P214
Volume 70 Issue 6, 114-121, June 2022

Original Article

CS Optimized Task Scheduling for Cloud Data Management

Mandeep Singh[1], Shashi Bhushan[2]

[1]*IKGPTU, Department of IT, Chandigarh Engineering College, Mohali, India.*
[2]*Department of Computer Science and Engineering, Patna, India.*

[1*]mandeepdevgan@gmail.com

Received: 03 April 2022 Revised: 23 May 2022 Accepted: 09 June 2022 Published: 27 June 2022

Abstract *- Task scheduling is the most recent networking technology in cloud computing. Among various technologies, the concept of Virtualization, dynamic sharing, delivering quality service, and load balancing are some of the most attractive ones that require high attention. While scheduling tasks and sharing applications, the most important challenge is to minimize execution time while maintaining the quality of service in terms of Service Level Agreement (SLA) and energy consumption. In the present paper, the authors proposed a Cuckoo Search Optimization to improve the local search strategy and schedule tasks in the cloud computing environment. This iterative search mechanism integrated efficient task scheduling with the neural architecture to achieve secure scheduling. The simulation analysis performed up to 1000 tasks for 100 user requests in terms of SLA violation, and energy consumption demonstrated the effectiveness of the proposed CS optimized, secure scheduling.*

Keywords *- Cloud Computing, Cuckoo Search, Modified Best Fit Decreasing, Neural Network, Scheduling.*

1. Introduction

Cloud computing is a growing concern due to its customers' wide range of services. It is a distributed system in which various devices are interconnected to utilize the resources provided by the service provider [1]. Energy efficiency plays a paramount role in such systems, and wider energy usage has led to the rise in cost and consumption for small, medium, and large-scale data centers. Consequently, Virtualization is an enabling technology that allows running the number of virtual machines (VM) on one physical machine (PM). VM is a middleware abstraction that separates the operating system from the physical infrastructure.

Additionally, the VM kernel is responsible for creating, running, and managing the multiple VMs on a pool of PM, as shown in Fig. 1. It is also called a scheduler that controls and manages the VM access to the PM. In the case of planned maintenance, one requires maintenance of software and hardware to update the important applications while VMs are not working, which avoids the system downtime. In case of a consolidation, VMs migrated on PMs having light load to avoid the hot-spot problem.

1.1. VM SCHEDULING

In a virtual environment, VM scheduling is a most challenging task [2] [3]. VM scheduling may be static or dynamic, depending upon the utilization of resources. Specifically, there are two major scheduling phases to allocate the resources in the cloud environment under the dynamic environment.

1. Allocation of Physical Resources (Infrastructure level)
2. Allocation of Virtual Resources (Application level)

In the first phase, resource allocation mainly depends upon the placement of the VM in the first attempt or changing its place, also termed as VM migration, as per available resources. During the first start-up time, the VM is allocated to the selected PM, and required resources are provided. In the later phase, changing or modification of VM takes place from one PM to another using the migration techniques. In the past, cloud computing practitioners have applied several migration techniques to place VM [4] efficiently.

Two types of migrations are usually observed. Live migration allows VM movement from one to another PM within the cloud system transparently while VMs are working [5] [6]. This technique provides various benefits, including load balancing, planned maintenance, and consolidation. In the case of load balancing, the resource utilization must be balanced during the migration process by avoiding hot-spot PMs while the system is working. In the case of planned maintenance, one requires maintenance of software and hardware to update the important applications while VMs are not working, which avoids the system downtime. In case of a consolidation, VMs migrated on PMs having light load to avoid the hot-spot problem and efficient utilization of resources [7].

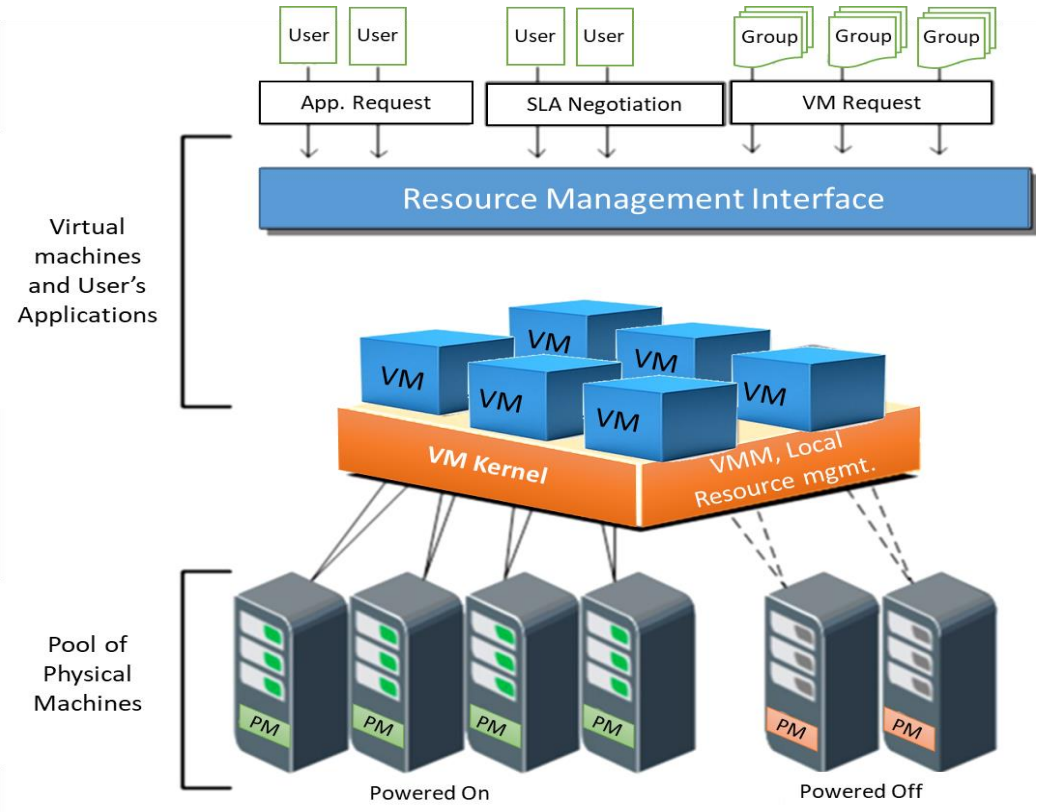

Fig. 1 Virtualization

Generally, there are different perspectives for the placement of VM to detect the PM for efficient VM migration. Some heuristics such as Best Fit Decreasing (BFD), First Fit Decreasing (FFD), Worst Fit Decreasing (WFD), and meta-heuristic techniques Particle Swarm Optimization (PSO), Cuckoo Search (CS), Artificial Bee Colony (ABC), and many more had been proposed in the literature for the efficient placement [8]-[10]. Reference [1] proposed the Modified Best Fit Decreasing (MBFD) algorithm for VM migration. The technique includes sorting VMs in decreasing order based on the current utilization of the CPU. Moreover, PM is allocated by a VM that shows the least use of power incurred due to reallocation. Such a heuristic technique is a part of the greedy algorithm that optimizes the VMs allocation but is limited to acquiring the optimal global solution. Therefore, researchers focused on power-aware and energy-aware techniques that control the power consumption and detect the underutilized PMs for balanced migration of VMs to achieve optimal solutions [11] [12].

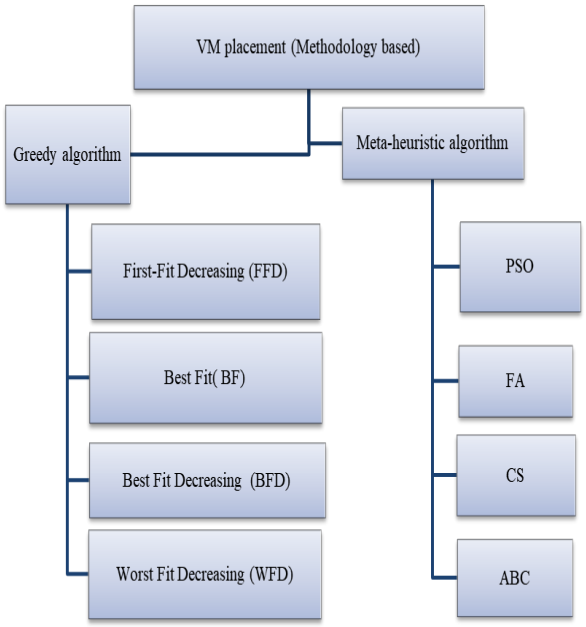

Fig. 2 VM placement and migration techniques using different methods

However, power techniques in conjunction with the optimization technique had been tested for VM placement [13]. However, these techniques cannot balance the resources that ensure minimum wastage of resources and power consumption. Such an imbalance may augment the problem

of resource wastage due to the use of a multi-dimensional resource. Therefore, developing a new VM scheduling and migration model is crucial to avoid the imbalance [14] [15].

Additionally, the main aim of this paper is to develop a secure scheduling algorithm using the metaheuristic technique. The study includes the Modified Best Fit Decreasing algorithm (MBFD) algorithm integrated with the Artificial Neural Network (ANN) classifier and further optimized using the Cuckoo Search technique. MBFD technique has been modified to avoid the limitation of the previous technique, and ANN classifier is used to avoid the problem of local optima but reallocation concept complex for some locations. Additionally, the CS technique has gained wide attention due to its fast convergence and fewer parameters in the real cloud environment [16].

1.2. Organization of Paper

The paper is organized as follows: part 1 introduces the VM scheduling and migration techniques in which the concept of Virtualization has been explained widely. The second part highlights the related work in which techniques used by the researchers and cloud service providers to solve the scheduling problem are highlighted. The third part highlights the methodology section in which VM scheduling using the MBFD and ANN has been defined well. The proposed technique is optimized using the CS algorithm to achieve the optimal solution. The next part demonstrates the results and discussion in which performance metrics have been defined and computed. Further, the proposed approach is compared with the existing study to validate the results. Finally, the paper is concluded in the last section.

2. Literature Review

In the past, researchers focussed on solving the scheduling problem using several optimizations and greedy algorithms [17]. Metaheuristic techniques have gotten wide attention in solving the combinatorial and VMs reallocation problem.

Reference [7] fostered a powerful energy-effective migration and VM consolidation technique. The study is based on developing an energy-efficient model that can limit energy utilization with ensured QoS. The authors proposed a strategy for a two-fold limit with usage to set off the movement of VMs. The Modified PSO (MPSO) strategy was used to solidify VMs so that they won't get stuck in nearby optima that have been an existing imperfection in most customary heuristic computations. The implemented technique compared with the well-known conventional heuristic method MBFD and calculation shows that physical nodes and VM migrations reduce, thus providing better energy effectiveness for distributed computing. The results show that MPSO had more load balance degrees than the MBFD technique.

However, Reference [18] focused on distributed computing, a combination of physical and virtualized resources for the clients. Fundamentally, better execution had been accomplished in task scheduling, organization, and optimization. This paper used the CS and Harmony Search (HS) algorithm to work on the scheduling problem. These two techniques were successfully consolidated to perform intelligent operations. The function integrated energy consumption, credit, and memory usage as indicated. At last, the outcomes of the proposed CHSA technique achieve the least expensive, least memory use, least energy utilization, and least pay in contrast to existing techniques.

Reference [19] discussed more than two techniques for dynamic host controller decision, Virtual affirmation and secure virtual improvement. A procedure developed an incredibly able handling condition for appropriate figuring wherein clients or various tenants are required from two or three sources. Through this model, the authors lined the overall response time (QoS) that should be insignificant. Further, this proposed set of rules has been assessed on the CloudSim test structure. The simulation outcomes show that the work could overcome the problems of existing methodologies. Reference [20] presented two algorithms for Scheduling and an Adaptive WFD VM Placement technique. The proposed work disseminated the resources with negligible power, memory, and request processing time. The former algorithm manages the collocated VMs that share the memory on the same machine. Then, this paper limits the number of relocations among the framework and power-saving, thus decreasing the power utilization VM migrations and performance degradation in the cloud framework by 28.1%, 57.77%, and 57.1%, respectively.

Reference [21] presented the VM allocation method using the enhanced-MBFD (E-MBFD) algorithm. The experimental results were the cross-approval of VMs allocated on PMs using the ANN classifier. Likewise, the proposed strategy was carried out to analyze the misleading allocations due to resource wastage. The experimental outcome shows that the proposed strategy improved regarding limited power utilized and fewer SLA violations in contrast to existing techniques. Reference [22] used the ANN classifier to optimize the resource wastage problem focussing on VM scheduling. The study estimates the future availability of data and resources and minimizes the VM migration. Moreover, the processing time taken by the data centers is also reduced with minimum response time to the users. Thus, VM migration was minimized with scalable and elastic solutions.

Reference [23] was focused on energy effectiveness using Virtualization to deal with resource usage problems. The heuristic and the metaheuristic-based, as described above unable to obtain the optimal global solutions. Researchers proposed a bio-inspired Frog leaping algorithm

to revamp the total execution time, the number of relocations, and energy utilization than the past work based on the PSO technique. The outcomes show that the simulation time reduction compared to PSO was 17.8%. Reference [24] proposed an ABC with Simulated Annealing (ABC-SA) based technique for efficient scheduling as per the priority of task size and request. This ABC-SA-based approach could revamp the efficiency with minimum searching time for VM placement. ABC-SA was executed and tested in the CloudSim environment, and the outcomes were further evaluated.

In recent times, one more research study has designed a scheduling policy that integrated the strengths of Heuristic Scheduling and Neural Network architecture. In the process, reinforcement learning was used to train the model. The analysis comprised a comparison against various optimization solvers and related policies. The parameters such as latency, execution time, and error were used to evaluate the work involved in the resource scheduling [Chen et al. 2022]. [25]

In another approach, the limitations such as slow convergence and high computation complexities of some swam optimizations were considered. In the light of such deficiencies, scheduling optimization was proposed using a combination of Particle Swarm Optimization and standard Whale Optimization. Finally, work exhibited a discrete resource allocation exhibiting a reduction of makespan up to 18% and 24% in addition to a reduction of energy consumption [Chhabra, 2022].[26]

A task scheduling was proposed that was based on the concept of a decision tree. The work also involved a comparative study conducted among existing algorithms. The simulation analysis showed that the designed tree-based algorithm improved resource allocation with significant load balancing. However, the evaluation did not consider power consumption and scalability parameters [Mahmoud, 2022].[29]

The state of techniques has been discussed, and it is analyzed that the conventional heuristic techniques cannot acquire the optimal solutions. Moreover, optimization techniques have been used in the literature, unable to balance the resources and scheduling problems.

3. Research Methodology

In the proposed work, the author had integrated CS as an optimization approach to identify the most suitable VMs that could be used for performing task scheduling in the existing cloud computing architecture. This information, together with the security level, is obtained using a neural network for final task scheduling. The performance of the designed optimization-based scheduling is analyzed for SLA, energy consumption, and observed delay.

3.1. The proposed technique using the MBFD

The resource analysis is the major activity that is performed before task scheduling. The proposed work initially implements MBFD for the tasks based on the energy constraints of the individual tasks. In case there are multiple outcomes based on one parameter, other parameters such as RAM and CPU utilization are also considered. The algorithmic steps of MBFD involved in the proposed work are as follows:

Algorithm 1: MBFD

1. $Input: data_{Task}$ // various tasks
   $data_{energy}$ // energy requirement of each task
2. foreach t in $data_{task}$
3. Assign : $min_{prority} \rightarrow max(energy)$ //highest energy required task is assigned min priority
4. Initalize variable for storage space: $storage_{space} = []$
5. Foreach p in $data_{energy}$
6. $Predicted_{priority} = estimation(data_{task}, data_{energy})$ //predicted priority based on energy
7. Compute mean value: $Priority_{mean} = mean\ (Predicted_{priority})$
8. if $(Predicted_{priority} < Priority_{mean})$
9. $storage_{space} = data_{task}$
10. $Priority_{min} = Predicted_{priority}$
11. End_{if}
12. If $(storage_{data} \neq [])$
13. assign: $data_{sorted} = (data_{task}, Predicted_{priority})$
14. End_{if}
15. End_{for}
16. End_{for}
17. Return: $data_{sorted}$ //sorted list of tasks with their priority value

In the next step, the data management is performed more wisely. The integration of the CS algorithm is done to obtain a sorted form of the data that is based on the energy requirement.

3.2. Cuckoo Search (CS) Optimization

It is a meta-heuristic technique based on the brood parasitism demonstrated by the cuckoo. It was initially explored by the researchers [28]. The sorted tasks are now re-analyzed for various parametric constraints, and an optimal priority list is designed based on the proficiency of cuckoos. This optimization has presented attractive solutions to numerous real-world applications. It exploits the upbringing behavior of cuckoos, i.e., laying color-pattern imitated eggs in other birds' nests. Here every single egg in the nest represents a solution, while the egg symbolizes a new potential solution.

Algorithm 2: Cuckoo Search (CS) Optimization

1. Input: $Data_{sorted}$ // sorted task priority list obtained using MBFD
2. Initialize CS parameters
   E_{size} // number of eggs symbolizing the property of sensor nodes
   E_{OT} // other eggs
3. $R = length(Data_{sorted})$ //length of optimized training data
4. Initialize variable:
5. $Data_{opt} = []$ //initialize optimized training data variable
6. $For\ each\ i\ in\ R$
7. $E_{current} = Data_{sorted}(i)$ // representing selected $task_{property}$ from $Data_{sorted}$
8. $E_{threshold} = average(Data_{sorted})$ // representing $threshold_{property}$
9. $F_{fit} = fit(E_{current}, E_{threshold})$
10. $Where\ Fitness\ is\ denoted\ as\ F_{fit} = \begin{cases} 1, & True \\ 0, & False \end{cases}$
11. $data_{opt} = CS(F_{fit}, Data_{sorted})$
12. End_{for}
13. $Return: data_{opt}$ //optimized task data

The above CS algorithm re-analyses the sorted tasks using its search mechanisms. This algorithm provides two types of task data. The "False" category represents one that needs a schedule, and the other that does not require the "True" category represents scheduling. Based on this information, two category data are passed to the neural network for training and predicting task scheduling. In other words, this information is then fed to the neural network for training and classification of tasks to avoid over-utilization and under-utilization scenarios. The steps used for the training and classifying of the tasks are given below.

Algorithm 3: Neural Network Architecture

1. Input: cat_{data} // category of task data
   $data_{opt}$ //optimized task data
2. Initialize Neural Network Parameters
   E_{num} // epochs count
   Itr_{num} // iteration or simulation rounds
   N_{num} // neuron count
3. $Performance\ parameters: MSE, Mutations, Gradient, and\ Validation$
4. $Techniques: Levenberg\ Marquardt$
5. $Data\ Division: Random$
6. $Foreach\ d\ in\ data_{opt}$
7. If $(data_{opt}\ belongs\ to\ over_{utilized})$
8. Assign $cat_1 == data_{opt}(d)$ // over utilized
9. If $(data_{opt_{train}}\ belongs\ to\ under_{utilized})$
10. Assign $cat_2 == data_{opt}(d)$// under utilized
11. Else
12. Assign $cat_3 == data\_opt\ (d)$ // representing optimal utilization
13. End_{if}
14. End_{for}
15. $Call\ neural\ network\ using\ Newff$
16. $net_{structure} = Newff\ (data_{opt}, cat_{data}, N_{num})$
    $net_{structure} =$
17. $train(net_{structure}, data_{opt}, cat_{data})$
18. $simulate\ net_{structure}$ with a $test_{data}$
    $Out = sim(net_{structure}, test_{data})$
    $Return: Out\ as\ load\ and\ resource\ utilization\ status\ of\ task$

The above neural network functions to identify the resource utilization status of VMs to which the task is scheduled. It first trains the system based on the optimized task scheduling data obtained with the implementation of MBFD followed by CS. The appropriate hosts were identified using the CS algorithm. This information is validated using a neural network so that the task is not assigned to an over-utilized host.

4. Results and Discussion

The performance of the proposed CS-optimized task scheduling algorithm is evaluated in terms of the number of SLA violations and energy consumption.

Table 1. Comparative Analysis of SLA violation using the proposed methodology and existing technique

Number of User Requests	MBFD + NN	Proposed MBFD + CS + NN	Nashaat et al [20]	Shalu & Singh [21]
10	0.00004	0.00002	0.000034	0.000028
20	0.00005	0.00004	0.000057	0.000046
40	0.00006	0.00005	0.0000684	0.000057
60	0.00006	0.000054	0.00007638	0.000059
80	0.000075	0.000063	0.0000855	0.000067
100	0.00009	0.00007	0.0001026	0.000078

The given table 1 depicts the comparative analysis of the proposed methodology with the existing techniques. The outcomes show that as there is an increase in the number of user requests, the number of SLA violations also increases. The proposed technique using the MBFD+CS+NN shows less number of violations for 10 requests while it is 0.000034 and 0.000028 using the existing technique [20] and [21], respectively. However, the technique with MBFD+NN shows 0.00004 violations of requests for 10 users. Consequently, the SLA violation for 40 users is 0.00005 and 0.00006 using MBFD+NN and the proposed methodology. The violation using the existing technique [20] and [21] is 0.00006 and 0.00005, respectively, for the same number of users.

Thus, it is clear that when the user request increases to 100, the SLA violation also increases, and it approaches 0.00009 and 0.00007 using the MBFD+NN and the proposed technique, respectively. However, the SLA violation for 100 requests using the [20] is 0.00010, and [21] is 0.000078. The average value obtained using the proposed technique is 0.00004, while using the MBFD+NN is 0.000062. Thus, the proposed technique outperforms the existing techniques.

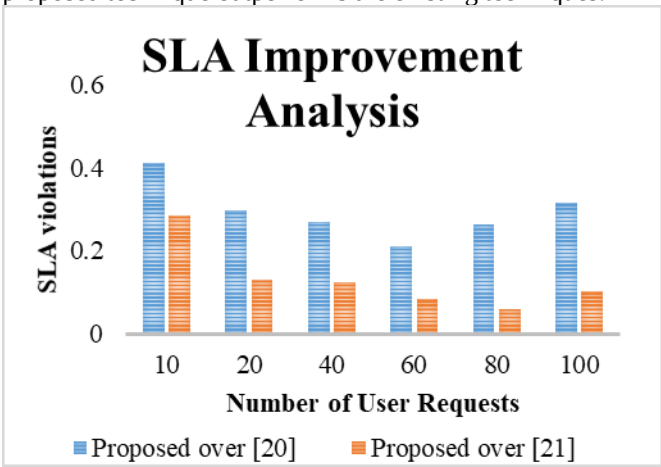

Fig. 3 SLA improvement analysis using the proposed over the existing techniques

Fig. 3 shows the improvement analysis of the proposed methodology over the existing techniques [20] and [21]. The proposed technique over the [21] for 20,40,60, and 80 has been improved by 13%, 12%, 8.4%, and 6%. The analysis results show that the proposed technique has improved by 41% and 28% compared to [20] and [21], respectively, for 10 requests. However, the number of user requests for 20, 40, 60, and 80 has been improved using the proposed methodology over the [20] is 29%, 27%, 21%, and 26%.

Similarly, as the number of requests approaches 100, the SLA violation over the [20] has been improved by 31% and [21] by 10%.

Table 2 compares the proposed methodology's comparative analysis with the existing techniques. It is clear that as the number of user requests increases, the energy consumed by the user also increases. The proposed technique shows minimum energy consumption in comparison to MBFD + NN approach and existing technique. The outcomes show that energy consumption also increases with the number of user requests.

Table 2. Comparative Analysis of Energy Consumption using the proposed methodology and the existing technique

Number of User Requests	MBFD + NN (kWh)	Proposed MBFD + CS + NN (kWh)	Nashaat et al [20]	Shalu & Singh [21]
10	32.41	19.41	34.6787	25.485
20	43.45	25.74	46.4915	29.648
40	49.37	29.84	52.8259	31.842
60	51.98	30.11	55.6186	35.845
80	53.14	33.32	56.8598	41.842
100	56.32	35.78	60.2624	43.235

The proposed technique using the MBFD+CS+NN shows minimum energy consumption for 10 requests while it is 34.6 kWh and 25.64 kWh using the existing technique [20] and [21], respectively. However, the technique with MBFD+NN shows 32.41 kWh energy consumed as some 10 user requests. Consequently, the energy consumption for 40 users is 49.37 kWh and 29.8 kWh using the MBFD+NN and the proposed methodology. The energy consumption using the existing technique [20] and [21] is 52.8 kWh and 31.8 kWh, respectively, for the same number of users.

Thus, it is clear that as the number of users requests increases to 80, then the energy consumption by users also increases, and it approaches 53.14 kWh and 33.32 kWh using the MBFD+NN and the proposed technique, respectively. However, the energy consumption for 100 requests using the [20] is 60.26 kWh, and [21] is 43.23 kWh. The average value of energy consumption using the proposed

approach is 29 kWh. However, the energy consumption using the MBFD + NN is 47 kWh. Thus, the proposed technique outperforms the existing techniques.

Fig. 4 shows the energy consumption improvement analysis of the proposed methodology over the existing techniques [20] and [21]. The proposed technique shows that energy consumption over the [20] for 20, 40, 60, and 80 has been improved by 44%, 43%, 45%, and 41%. The analysis results show that the proposed technique has been improved for 100 requests by 40% and 17% compared to [20] and [21], respectively, for 10 requests. However, the number of user requests for 20, 40, 60, and 80 has been improved using the proposed methodology over the [21] is 13%, 6.2%, 16%, and 20%. Similarly, as the number of requests approaches 100, the energy consumption over the [20] has been improved by 40% and [21] by 17%.

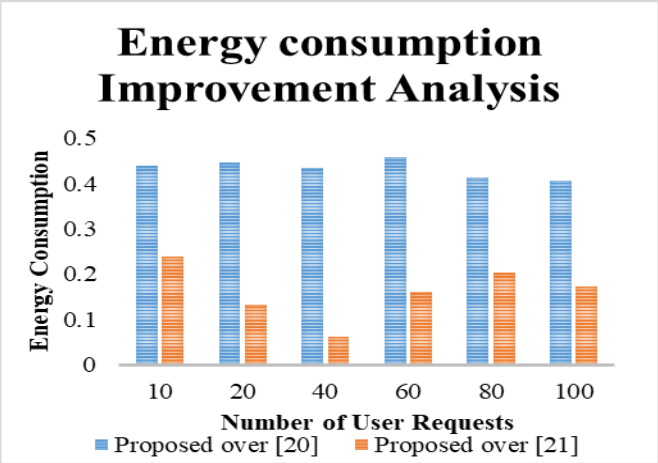

Fig. 4 Energy Consumption improvement analysis using the proposed over the existing techniques

5. Conclusion

The present research work presents a CS-based scheduling technique for the secure placement and migration of VMs. The proposed methodology is based on using the MBFD technique for the best fit, and then a Neural Network classifier has is used to classify the VMs. The proposed study optimized using the CS, a robust technique to optimize the scheduling tasks. The experimental results have been evaluated using energy consumption and SLA violation. The outcomes show that as the number of user requests increases, SLA violation also increases, and the machines consume more energy. However, the proposed technique shows less number of violations and minimum energy consumed by 10 users' requests. The simulation results show that the proposed approach's energy consumption has been improved by 43%, and SLA violation is improved by 29%. Thus, prominent results were obtained by integrating the MBFD with CS and NN.

References

[1] A. Beloglazov, C. Abawajyb and R. Buyya, Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Center for Cloud Computing, Future Generation & Computer Systems. 28(5) (2012) 755–768.

[2] B. Wang, L. Fagui and L. Weiwei, Energy-Efficient VM Scheduling Based on Deep Reinforcement Learning, Future Generation Computer Systems. 125 (2021) 616-628.

[3] C. Sudhakar and T. Ramesh, Energy-Efficient VM Scheduling and Routing in a Multi-Tenant Cloud Data Center, Sustainable Computing, Informatics and Systems. 22 (2019) 139-151.

[4] C. D. Wei, F. L. ChunhuaGu, C. Yaohui, R. Ulysse, L. Xiaoke, and W. Geng, DFA-VMP: An Efficient and Secure Virtual Machine Placement Strategy Under Cloud Environment, Peer-to-Peer Networking, and Applications. 11(2) (2018) 318-333.

[5] A. Gupta, and S. Namasudra, A Novel Technique for Accelerating Live Migration in Cloud Computing, Automated Software Engineering. 29(1) (2022) 1-21.

[6] N. Mostafa, A. Ibrahim, and H. A. Ali, Optimization of Live Virtual Machine Migration in Cloud Computing: A Survey and Future Directions, Journal of Network and Computer Applications. 110 (2018) 1-10.

[7] H. Li, G. Zhu, C. Cui, H. Tang, Y. Dou, and C. He, Energy-Efficient Migration and Consolidation Algorithm of Virtual Machines in Data Centers for Cloud Computing, Computing. 98(3) (2016) 303-317.

[8] B. B. Naik, D. Singh, and A. B. Samaddar, FHCS: Hybridised Optimization for Virtual Machine Migration and Task Scheduling in a Cloud Data Center, IET Communications. 14(12) (2020) 1942-1948.

[9] H. O. Salami, A. Bala, S. M. Sait, And I. Ismail, The Journal of Supercomputing, An Energy-Efficient Cuckoo Search Algorithm for Virtual Machine Placement in Cloud Computing Data Centers. 77(11) (2021) 13330-13357.

[10] K. Tuli, and A. Kaur, Hybridization of Harmony and Cuckoo Search for Managing the Task Scheduling in Cloud Environment, in Proceedings of Data Analytics and Management. (2022) 749-761.

[11] D. M. Zhao, J. T. Zhou, and K. Li, An Energy-Aware Algorithm for Virtual Machine Placement in Cloud Computing, IEEE Access. 7 (2019) 55659-55668.

[12] I. Leila and H. Materwala, Energy-Aware VM Placement and Task Scheduling in Cloud-Iot Computing: Classification and performance Evaluation, IEEE Internet of Things Journal. 5(6) (2018) 5166-5176.
[13] K. Balaji, P. Sai Kiran, and M. Sunil Kumar, Power-Aware Virtual Machine Placement in IAAS Cloud Using Discrete Firefly Algorithm, Applied Nanoscience. (2022) 1-9.
[14] K. Vincent, E. Madelaine, and F. Hermenier, Scheduling Live Migration of Virtual Machines, IEEE Transactions on Cloud Computing. 8(1) (2017) 282-296.
[15] Ahmad and R. Wasim, A Survey on Virtual Machine Migration and Server Consolidation Frameworks for Cloud Data Centers, Journal of Network and Computer Applications. 52 (2015) 11-25.
[16] E. Barlaskar, J. S. Yumnam, and B. Issac, Enhanced Cuckoo Search Algorithm for Virtual Machine Placement in Cloud Data Centers, International Journal of Grid and Utility Computing. 9(1) (2018) 1-17.
[17] Z. Zhong, K. Chen, X. Zhai, and S. Zhou, Virtual Machine-Based Task Scheduling Algorithm in a Cloud Computing Environment, Science and Technology. 21(6) (2016) 660-667.
[18] K. Pradeep, and T. Prem Jacob, A Hybrid Approach for Task Scheduling Using the Cuckoo and Harmony Search in a Cloud Computing Environment, Wireless Personal Communications. 101(4) (2018) 2287-2311.
[19] S. N. Raghavendra, K. M. Jogendra, and C. C. Smitha, A Secured and Effective Load Monitoring and Scheduling Migration VM in Cloud Computing, In IOP Conference Series: Materials Science and Engineering. 981(2) (2020) 22-39.
[20] H. Nashaat, N. Ashry, and R. Rizk, Smart Elastic Scheduling Algorithm for Virtual Machine Migration in Cloud Computing, The Journal of Supercomputing. 75(7) (2019) 3842-3865.
[21] D. Shalu, and D. Singh, Artificial Neural Network-Based Virtual Machine Allocation in Cloud Computing, Journal of Discrete Mathematical Sciences and Cryptography. 24(6) (2021) 1739-1750.
[22] A. Radhakrishnan, and V. Kavitha, Energy Conservation in Cloud Data Centers by Minimizing Virtual Machines Migration through an Artificial Neural Network, Computing. 98(11) (2016) 1185-1202.
[23] G. Kumar, and P. Vivekanandan, Energy-Efficient Scheduling for Cloud Data Centers Using Heuristic-Based Migration, Cluster Computing. 22(6) (2019) 14073-14080.
[24] B. Muthulakshmi, and K. Somasundaram, A Hybrid ABC-SA-Based Optimized Scheduling and Resource Allocation for Cloud Environment, Cluster Computing. 22(5) (2019) 10769-10777.
[25] Chen, Z., Wei P & Li Y, Combining Neural Network-Based Method with Heuristic Policy for Optimal Task Scheduling in Hierarchical Edge Cloud, Digital Communications and Networks. (2022).
[26] Chhabra A, Huang K. C, Bacanin N & Rashid T. A, Optimizing Bag-of-Tasks Scheduling on Cloud Data Centers Using Hybrid Swarm-Intelligence Meta-Heuristic, The Journal of Supercomputing. 78(7) (2022) 9121-9183.
[27] Mahmoud H, Thabet M, Khafagy M. H & Omara F. A, Multiobjective Task Scheduling in Cloud Environment Using Decision Tree Algorithm. IEEE Access. 10 (2022) 36140-36151.
[28] Yang, Xin-She, and S. Deb, Cuckoo Search: Recent Advances and Applications, Neural Computing and Applications. 24(1) (2014) 169-174.