International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Dynamic Load balancing Approaches with Optimal Virtual Machine Migration in Cloud Environments

Rudresh Shah[1*], Suresh Jain[2], Kailash Chandra Bandhu[3]

Submitted: 29/01/2024 **Revised**: 07/03/2024 **Accepted**: 15/03/2024

Abstract: In recent years, Cloud Computing has become increasingly appealing to service providers seeking to run applications on large data centres, primarily due to the advantages of highly available hardware, on-demand provisioning, and pay-as-you-go models. This technology harnesses the power of virtualization, which allows for the consolidation of multiple Virtual Machines (VMs) onto a minimal number of servers. By employing dynamic VM provisioning, VM consolidation, and strategically switching servers on and off as needed, data centres can maintain the desired Quality-of-Service (QoS) while achieving greater server utilization and energy efficiency. In our proposed work, we focus on managing the inter-relationship between energy consumption, the number of VM migrations, SLA (Service Level Agreement) violations, and application performance. Our approach tackles the issue of over-utilized servers by migrating the most suitable VMs to appropriate destination servers. To accomplish this, we have devised VM selection and VM placement strategies. Additionally, for overload detection, we have utilized the exponential smoothing technique. Our chosen platform for implementing these approaches is the cloudsim simulator. The results of our study demonstrate significant benefits. Specifically, we observed a reduction in energy consumption of up to 17.57%, a decrease in the number of VM migrations, and overall improvements in application performance across various scenarios.

Keywords *cloud computing; virtual machine consolidation; energy consumption; virtual machine migration; hotspot mitigation.*

1. Introduction

In recent years, Cloud computing has emerged as a transformative paradigm for large-scale parallel and distributed computing. It offers convenient and on-demand network access to a shared pool of configurable computing resources, including networks, servers, storage, applications, and services [1]. These resources can be rapidly provisioned and released with minimal management effort, making it an attractive option for service providers due to its pay-as-you-go policy and the absence of upfront capital investment. Consequently, distributed large-scale data centers, comprising thousands of hosts, have become the preferred infrastructure for offering services.

Cloud Computing is a rapidly emerging technique that provides online computing resources, storage, and infrastructure. Its main characteristics include on-demand self-service network access, resource pooling, and rapid elasticity of service availability. Cloud Computing has four main deployment models: Private Cloud, Public Cloud, Community Cloud, and Hybrid Cloud. Additionally, it offers three different service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). One of the significant concerns in Cloud Computing is load balancing. Ensuring load balancing is crucial to avoid scenarios where tasks are waiting in a queue for service at one resource, while another capable idle resource remains unused. Load balancing algorithms aim to minimize such situations as much as possible [2]. Swarm intelligence techniques have demonstrated their effectiveness in solving load balancing problems. They offer promising solutions for achieving load balancing by minimizing the load difference between the heaviest and lightest nodes. This paper proposes a distributed swarm intelligence-inspired scheduling and load balancing algorithm called Ant Colony Optimization (ACO).

However, the operation of these data centers comes with significant challenges related to energy consumption. The substantial number of hosts and the inefficient load distribution on servers lead to excessive heat generation by overloaded servers, necessitating more cooling, which, in turn, results in high carbon dioxide emissions [3]. Beyond the operational costs, the considerable energy consumption poses reliability concerns, as the failure rate of hosts doubles for every 10-degree increase in temperature [4]. Therefore, the critical task at hand is to find ways to reduce energy consumption without compromising performance while managing cloud data center operations effectively. Virtualization Technology has revolutionized server capabilities by allowing multiple Virtual Machines (VMs) to run concurrently on a single server. VM consolidation is a widely used approach to reduce energy costs, achieved through live VM migration operations, which balance the workload and

1,2,3 Medi-Caps University, Indore, India
** Corresponding Author Email: rudresh.shah@medicaps.ac.in*

optimize computing resources by relocating VMs from one server to another. This process consolidates VMs onto a smaller number of active hosts, increasing the number of inactive hosts. Leveraging virtualization capabilities, we can dynamically scale up or down computing resources to accommodate fluctuating workloads effectively. During non-overloaded periods, we can add more VMs, and during underutilized times, we can consolidate VMs, minimizing active hosts and even turning off idle hosts. Notably, idle hosts consume a significant amount of power, with completely idle hosts wasting over 70% of their peak load [5]. Hence, powering off idle hosts results in considerable power savings.

However, virtualization also introduces challenges in resource management due to resource sharing among multiple VMs on a single server [6]. This sharing can lead to performance uncertainties for VMs, primarily caused by server overload [7]. Thus, striking a balance between avoiding overload for better performance and dynamically consolidating VMs for greener computing becomes crucial. Efficiently managing cloud resources poses additional difficulties, as interactive service applications often generate dynamic and non-deterministic resource demands due to frequently changing workloads. Active VM consolidation can lead to performance degradation in such scenarios. As a consequence, there is a trade-off between energy efficiency and performance, which can lead to Service Level Agreement (SLA) violations. SLAs are agreements between service providers and consumers to ensure reliable Quality-of-Service (QoS).

To achieve server consolidation, live migration techniques [8] are employed, creating a direct relationship between VM consolidation and overload avoidance, ultimately leading to improved performance while adhering to SLAs. Live migration allows the seamless movement of a running VM or application between different servers without disrupting client or application connections. This approach enhances performance and reduces energy consumption by effectively addressing underutilization and overutilization issues. Numerous research efforts have been dedicated to handling VM migration, considering that each migration introduces some performance degradation, leading to increased SLA violations and operational costs. Therefore, careful consideration is required to determine which VMs should be migrated and where they should be migrated.

The primary objective of our work is twofold: reducing energy consumption and minimizing the number of VM migrations while maintaining low SLA violations. Excessive VM migrations can cause network congestion and add overhead for tracking VM migration information. To address these challenges, we employ future forecasting methods for load prediction, enabling us to proactively avoid server overload problems. This predictive capability also aids in deciding whether a VM migration should be performed, helping to minimize the overall number of migrations. Throughout this process, we ensure that performance remains at acceptable levels. Our main areas of focus are as follows:

- Proposing VM selection approaches based on VM's CPU load and RAM size to handle over-utilized servers effectively.
- Introducing VM placement approaches based on VM's CPU load to optimize server utilization efficiently.

2. Related Work

Over the past few years, researchers have been increasingly concerned about the substantial energy consumption of cloud data centers. As a response, numerous energy-aware and VM consolidation algorithms have been developed. Within this context, two critical issues have been addressed: The first issue is overload avoidance, which involves the use of load prediction algorithms to forecast future resource demands. These predictions are utilized to apply migration techniques, ensuring that server loads remain below the upper threshold of utilization. By proactively managing resource allocation, the aim is to prevent servers from becoming overloaded and optimize their performance. The second issue pertains to VM consolidation, which aims to reduce the number of active hosts by migrating VMs from underutilized servers. By consolidating VMs onto fewer hosts, the data center can achieve better resource utilization and energy efficiency.

In summary, researchers have focused on developing energy-aware and VM consolidation algorithms to tackle the challenges of high energy consumption in cloud data centers. These algorithms address overload avoidance through load prediction and migration techniques, as well as VM consolidation to enhance resource utilization. [9] utilizes monitoring of VMs and hosts' CPU and memory. The authors propose a fixed threshold value to limit resource utilization. [10] introduces a resource management system based on virtualization to reduce energy consumption while maintaining QoS. [11] presents push and pull load balancing algorithms that involve VM migrations. [12] proposes various energy-efficient VM-to-host mapping methods, including three VM selection policies: "minimization of migration," "highest potential growth," and "random choice." In later work, [7] addresses continuously changing CPU utilization thresholds due to varying workloads, introducing adaptive techniques like Median Absolute Deviation (MAD) and Inter Quartile Range (IQR). For VM placement, [13] suggests strategies based on overloaded and under-loaded

PMs using push and pull approaches. [14] decides VM migrations and destination selection based on forecasting heuristics and smoothening techniques.

In [15], the author introduces two strategies for automated live migration of virtual machines, based on overloaded and under-loaded Physical Machines (PMs). The first strategy, known as the push strategy, involves PMs with loads exceeding a threshold acting as sellers, while the VMs resident on these PMs are auctioned items. Information shared with buyers includes CPU load and memory usage. The second strategy, called the pull strategy, features under-loaded PMs acting as sellers, providing information about available free resources. In [13], the authors propose a method for deciding resource reallocation of virtual machines, addressing the dependency problem present in application migration. Instead of using fixed threshold methods, they define a stochastic model based on cumulative sum. This stochastic method effectively eliminates unwanted VM migrations caused by slight changes in CPU load. For destination host selection, a best-fit method is employed.

In [14], the author first determines whether VM migration is required based on forecasting heuristics. If migration is deemed necessary, an appropriate destination is selected. Both VM migration and destination selection employ smoothing techniques. VM migration occurs in two main cases: when the server load is above the dynamic threshold and when server loads fall below the lower threshold. In the latter case, all VMs from the underutilized server need to be migrated. If server utilization exceeds the upper threshold, the predicted load is checked, and if it is also high, VM migration takes place. For destination selection, the authors propose a novel approach that considers the host's future load to ensure it remains unburdened when the migrated VM resumes operation. The Ant Colony Algorithm (ACO) was first proposed by Marco Dorigo and his colleagues in 1992, inspired by the behaviour of real ants. When searching for food, ants can efficiently discover the path between their nest and the food source. During this search, ants wander randomly and leave chemical substances called pheromones on the ground during their return trip. Other ants can then follow these pheromone trails to find the path to the food source and return to the nest. By following the pheromone trails with the highest density, ants can reach the food sources effectively, allowing for indirect communication and enabling them to find the shortest paths.

In [16], the proposed technique builds upon ACO, aiming to identify overloaded nodes in the shortest time and balance the load among nodes while maximizing resource utilization. In [17], an initial heuristic algorithm is applied to modify Ant Colony Optimization for service allocation and scheduling mechanisms in cloud systems. This modification supports minimizing the Makespan of the cloud system services. In [18], [19], a cloud task scheduling policy based on Load Balancing Ant Colony Algorithm is proposed. The approach aims to balance the entire system load while minimizing the Makespan of given tasks. In [20], an ACO-based approach is developed as an effective load balancing algorithm capable of maximizing or minimizing different parameters. In [21], a technique based on ACO is proposed for redistributing overloaded nodes based on a Threshold value. If the load on the current node is less than the threshold, ants will search among the available nodes, and their movement will be restricted to one direction.

Our approach makes use of the ant colony optimization technique to estimate the VM selection of the host which helps us to mitigate the problem of unnecessary VM migrations. In our work, we represent the interrelationship between energy consumption, the number of VM migration, average SLA violation, and Performance degradation

scheduling framework. In the context of cloud computing, the task scheduling problem becomes more intricate due to the need for additional steps such as VM migration, server consolidation, and ensuring compliance with Service Level Agreements (SLAs). To address these complexities, a framework is designed and developed for server consolidation, as depicted in Figure 1. Interactive applications in the cloud are dynamic, with varying resource requirements and loads. The primary objectives of the research discussed in this chapter are to devise approaches that efficiently utilize resources and minimize energy consumption. Achieving these goals necessitates VM migration.

However, VM migration poses challenges as it introduces system overheads, including reallocation of VMs and the need to keep track of VM migration information. Additionally, an excessive number of VM migrations can lead to network congestion. In this research work, comprehensive solutions are developed to address all three aforementioned problems: (i) determining when to migrate VM(s), (ii) identifying which VM(s) to migrate, and (iii) determining where to migrate VM(s) to optimize resource utilization and minimize energy consumption.

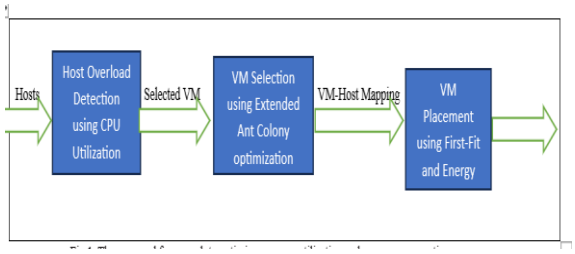

Fig 1. The proposed framework to server consolidation.

The proposed framework addresses all three problems and presents an illustrative representation of server consolidation approaches in Figure 1. To address the first problem, an algorithm for host overload detection is proposed, which involves determining an adaptive threshold for CPU utilization and implementing efficient load forecasting techniques. These forecasting techniques help predict resource utilization patterns, enabling timely decisions about VM migration. For the second problem, VM selection algorithms are introduced to choose one or more VM(s) for migration from overloaded servers. VM selection is based on the CPU and RAM usage of VMs on the over-utilized servers. Lastly, the framework addresses the third problem through the introduction of VM placement algorithms. These algorithms focus on effectively placing VMs in suitable destination hosts, ensuring optimized resource utilization and minimizing energy consumption.

3. Proposed Work

In accordance with our problem statement, our approach involves several steps. Initially, we determine the upper threshold of CPU utilization to establish a basis for identifying potential host overloads. We proceed to select the appropriate VM that will be migrated from an overloaded host. To achieve this, we introduce Ant colony optimization VM selection algorithm in the subsequent subsection. we present the VM placement algorithm, which is responsible for choosing the destination host for the selected VM from the previous step.

3.1. Host overload detection

To achieve an adaptive and dynamic utilization threshold for our system, we recognized that a static threshold is inadequate in a diverse and fluctuating workload environment where various types of applications share cloud resources, each with different utilization demands. To address this challenge, we adopted the MAD technique (median of the absolute deviations from the data set median) as discussed in the following sub-section. The MAD technique serves as a statistical dispersion measure and proves to be more effective in handling outliers in the data set compared to the standard deviation. By leveraging the MAD technique, our approach dynamically adjusts the utilization threshold based on the resource demands of the workload. This ensures that the threshold remains relevant and responsive to the varying workload conditions, allowing for more accurate and efficient performance in our system.

The MAD, denoted as MAD(X), is computed from the univariate data set $X=\{x_1,x_2,\ldots\ldots x_n\}$

$$MAD(X)= \text{median}\ (|x_i - \text{median}(X)|) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ (1)$$

The upper utilization threshold, denoted as uT, is determined by the formula:

$$uT= \text{median}(X) - s.MAD(X)$$

Here, $s \in R+$ is a safety parameter that influences the system's VM consolidation strategy. Smaller values of s lead to reduced energy consumption but may result in higher SLA violations during VM migration handling. we introduce our overload detection algorithm, which takes into account the upper threshold of the host's CPU utilization. To calculate the upper threshold, we utilize the MAD technique. This allows us to evaluate the effectiveness and accuracy of MAD technique in estimating the load of the host's CPU utilization, providing valuable insights into the performance of our overload detection algorithm.

Algorithm 1: Host Overload Detection Algorithm

1. **Input**: hostList, **Output**: migration decision
2. Set flagP = flagF = false
3. For each host in hostList do
4. Calculate host.currentCPUutil as host.getCurrentRequestedMIPS() / host.getTotalMIPS()
5. Obtain hostUtilData[] containing the historical CPU utilization data for the host
6. Calculate the upperThreshold (Tu) as $1-s\times MAD$
7. If host.currentCPUutil > upperThreshold (Tu), then
8. Set flagP = true
9. If hostUtilData[].size() < 10, then
10. If flagP = true, then
11. Add the host to currentOverUtilHost list (a list to store currently overloaded hosts)
12. If flagP == true and flagF == true, then
13. Return true (indicating that migration decision is needed)
14. Else
15. Return false (indicating that migration decision is not needed)

The algorithm functions as follows:

In line 2, two flags are defined: flagP for the present CPU utilization and flagF for the future CPU utilization. In line 4, the current CPU utilization of the host is calculated. In line 5, the historical CPU utilization data is obtained from the utilization history. In line 6, the host's upper threshold is calculated using the MAD technique. This threshold is

used to determine if the current or future CPU utilization exceeds the acceptable limits. If the current utilization of the host is higher than the calculated upper threshold, the flagP is set to true, indicating that the host is currently experiencing an overload (lines 7 to 8). We consider this host to be currently overloaded. If the historical utilization data for the host contains less than 10 data values (10 being a suitable value that provides sufficient data for the MAD calculation; other values such as 8, 12, 14, etc., can also be chosen), the currently overloaded host is added to the "currently over-utilized host" list. This step is necessary because the MAD calculation relies on the previous utilization history to detect host overloads (lines 9 to 11). However, at this point, no VM migration is performed yet, as it is only a preliminary step in the algorithm. Using this algorithm, we also exclude the current overutilized host from probably destination host list. That helps us to avoid unnecessary VM migrations.

3.2. Extended Ant Colony optimization based VM selection approach (EAC)

In the proposed load balancing approach with Extended Ant Colony Optimization (EAC), the main procedure can be broken down into three distinct steps, which are explained as follows:

3.2.1 Initializing the pheromone

In this step, the system initializes the EAC algorithm with necessary parameters and constructs the problem-specific graph representation. The graph represents the cloud environment, where nodes represent virtual machines (VMs) and edges represent the communication links or distances between VMs. Each VM is associated with a pheromone value that influences the ants' decision-making during the optimization process. Additionally, the algorithm sets other ACO-specific parameters, such as the number of ants, evaporation rate, and heuristic information. The initial amount of pheromone is assumed to be a small positive constant $x_{mn}(t) = 0$.

3.2.2 How Ants choose VMs

During this step, multiple ants traverse the graph to construct candidate solutions for the load balancing problem. Each ant starts from a randomly chosen VM and follows a probabilistic decision-making process to determine its next move. The decision is influenced by both the pheromone information and the heuristic information, which represents the desirability of moving from the current VM to another one. The ants continue their movement, constructing feasible solutions by visiting VMs while adhering to constraints and balancing the load. In the conventional Ant Colony Optimization (ACO) approach, the probability of selecting the next operation considers pheromone trails and edge weights. However, there is a recognized need to incorporate machine load as a crucial factor in this probability calculation. To address this requirement, the Load Balancing ACO introduces an additional parameter - Machine load. The traditional ACO algorithm does not take into consideration the machine load factor, which is crucial for minimizing idle time and achieving load balancing in the system. When a machine assigned to a particular operation is heavily loaded, the probability of completing that operation decreases, leading to a preference for other operations assigned to less burdened machines. In the proposed Load Balancing ACO, the ant determines the next node to visit based on equation 1, which incorporates the machine load as a significant parameter in the decision-making process.

$$\sum P(m,n) = \begin{cases} \frac{x(m,n)^{\alpha} * y(m,n)^{\beta} * z(m,n)^{\gamma}}{\sum u \epsilon S_k \; x(m,u)^{\alpha} * y(m,u)^{\beta} * z(m,u)^{\gamma}} & if\ n \epsilon S_k \ldots\ldots \\ 0, & otherwise \end{cases} \quad \ldots\ldots\ldots (1)$$

In the equation, x(m,n) represents the pheromone trail, y(m,n) represents the edge weight and z(m,n) represents the total execution time of the machine associated with a specific operation. An array is utilized to store the total execution time for each machine. When a machine is already busy executing an operation, selecting the next operation to be executed on the same machine can lead to increased idle times for other machines. To mitigate this issue and reduce idle time, the machine load parameter is taken into account. Consequently, machines with lower execution times are favoured, resulting in an overall reduction of idle time across the system. The parameters α, β and Γ determine the relative importance of the pheromone level compared to the edge weight. By adjusting the values of α, β and Γ the node with an edge of lower weight and higher pheromone levels is favoured in the path selection process. Once the path is selected, the ants deposit pheromone according to equation 2.

3.2.3 Pheromone Update

After all ants complete their tours and construct solutions, this step involves updating the pheromone values on the graph. The pheromone update strengthens the paths taken by the ants that led to good solutions and weakens the less successful paths. Based on the constructed solutions, the algorithm performs load transfer between VMs to achieve better load balancing. The VMs' load is adjusted according to the solutions found by the ants, effectively redistributing the workload.

$$x(m,n) = (1-\rho) * x(m,n) + \rho * \Delta x(m,n) \ldots\ldots\ldots (2)$$

Where ρ= decay, $0 < \rho < 1$ and Δx(m,n) is calculated by formula (3).

$$\Delta x(m,n) = \begin{cases} G_{gb} & ,if(m,n) \in goble\ best\ tour \\ 0 & otherwise \end{cases} \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

The ant system possesses a crucial characteristic known as pheromone evaporation, where the deposited pheromone gradually decreases over time. This property plays a significant role in preventing premature convergence to a sub-optimal solution. By allowing pheromone levels to decrease, the system maintains exploration opportunities, enhancing the chances of finding better and more optimal solutions instead of getting stuck in local optima. These three steps constitute the core process of the load balancing approach with ACO. The algorithm iterates through these steps multiple times until a satisfactory load-balanced state is achieved in the cloud environment. Through the intelligent movement of ants and the dynamic pheromone updates, the ACO-based approach effectively optimizes the distribution of tasks among VMs, improving the overall performance and resource utilization in the cloud system.

3.3. VM placement algorithm

We conducted a comparison of our results with two VM placement algorithms in the CloudSim simulator. The first algorithm is the Power Aware Best-Fit Decreasing (PABFD) algorithm, as simulated by Calheiros et al. (2011). The primary focus of PABFD is on the energy factor, and it allocates the host to a migratable VM whose energy difference after allocation is minimized. However, this approach may lead to a host overload problem in the near future, after each time period. The second algorithm used in our work is the First-Fit algorithm, which we had previously proposed (Choudhary et al., 2016). The First-Fit algorithm maps the VM to a host by finding the first suitable host in the host list. This algorithm significantly reduces the time complexity to O(mn), where m is the number of feasible destination hosts (equation (4)) and n is the number of migratable VMs. In some cases, the First-Fit algorithm performs better than PABFD because it allocates the host without concentrating on energy factors, which reduces the probability of host overload in the near future. While PABFD may have benefits in terms of energy efficiency, the First-Fit algorithm's focus on immediate suitability can lead to improved overall performance by preventing host overload issues.

$$FDH = (Total\ host - current\ overloaded\ host) \ldots\ldots (4)$$

Where FDH is feasible destination host. We proposed the destination host CPU utilisation aware VM placement algorithm 2.

Algorithm 2: Host Utilization-Based Algorithm

1. **Input:** hostList, VMList **Output:** Allocation map of VMs
2. **For each** VM in VMList do
3. Set **minPower** to double.MAXVALUE
4. Set **allocatedHost** to NULL
5. Set **hostSelect** to NULL
6. **For each** host in getHostList() do
7. if (currentOverloadHost.contains(host)) then
8. continue to the next host
9. if (host.isSuitableVM(VM)) then
10. if (getUtilOfCpuMips(host != 0 && isHostOverUtilAfterAllocation(host,VM)) then
11. continue to the next host
12. Increment totalhost
13. hostu=hostu+ host.getUtilOfCpu()
14. hostavg=hostu/totalhost
15. utildiff=Double.MAX_VALUE
16. for (host : getHostList()) do
17. repeat step 7-8
18. if (host.isSuitableForVM(VM)) then
19. repeat step 10-11
20. hostutil=host.getUtilOfCpu()
21. if (0<(hostutil-hostavg)<utildiff) then
22. utildiff=|hostutil-hostavg|
23. Allocation.add(VM,allocatedHost)
24. if(allocatedHost != NULL) then
25. call First-Fit or EABFD
26. return Allocation

The algorithm is called for each migratable VM, as shown in lines 2 to 26. For each defined host (lines 6 to 14), the algorithm calculates the average CPU utilization of the hosts. It checks whether the host is a feasible destination for the VM (lines 7 to 8). If the host is suitable in terms of CPU utilization, RAM, and bandwidth, it further checks whether allocating the VM will cause the host to overload. If it will, the algorithm moves to the next probable destination host; otherwise, it calculates the CPU utilization of that host (lines 9 to 13). The algorithm calculates the average CPU utilization of all suitable destination hosts (line 14). To find the mapping of VMs to destination hosts, the algorithm once again accesses each host (lines 16 to 25). The algorithm repeats the 2nd step (lines 17 to 21) to further explore suitable destination

hosts for the VM. It calculates the current utilization of the host's CPU (line 22). Next, the algorithm searches for a host whose CPU utilization is approximately close to the average utilization calculated in step 3, with a positive difference. It then allocates that host to the migratable VM. If the algorithm fails to find any suitable destination host in the previous step, it resorts to calling either the First-Fit or PABFD algorithm for the mapping of migratable VMs to hosts.

Our proposed VM placement algorithm (Algorithm 2) effectively reduces the number of overloaded hosts by allocating migratable VMs to hosts with average utilization, thus optimizing the distribution of workload. Additionally, it minimizes the need for migrations, leading to enhanced resource utilization and improved overall performance in the cloud environment.

4. SIMULATION And RESULT

CloudSim 3.0 [22] serves as the simulation tool employed in this study. The simulation is conducted under the following scenario: The tasks to be executed are independent of each other. Tasks vary in computational sizes, with the length of each task presented in Millions of Instructions (MI). The experiments are carried out on 20 tasks and 75 virtual machines (VMs). The cloud simulator s configured with the parameter settings outlined in Table I.

Table I Parameters Setting of CloudSim.

Type of Entity	Parameters	Values
Task (cloudlet)	Length of task	1000-20000 (MI)
	Total Num of Task	20
Virtual Machine	Total Number of VMs	75
	MIPS	500-2000
	VM memory	256-2048
	Bandwidth	70-100
	Number Of PEs Requirement	01-Apr
Datacenter	Number of datacenter	1
	Number of Host	02-Jun
	Vmscheduler	Space_shared and Time_share

Table II shows selected parameters of Extended ACO and ACO taken into experiments for simulation.

Table II Parameters Setting of CloudSim.

Parameter	Alpha	Beta	Gamma	Rho	G	t_{max}
EACO	0.1	0.9	5	0.4	100	100
ACO	0.2	1	0	0.4	100	100

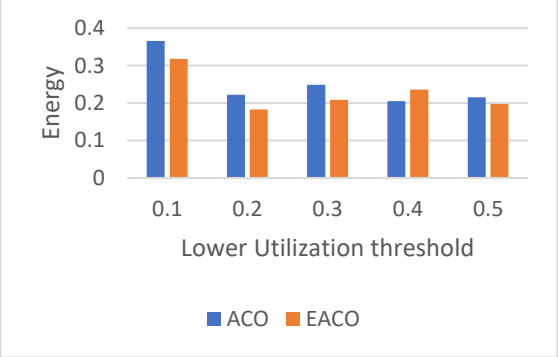

Fig 2 Energy Vs Threshold

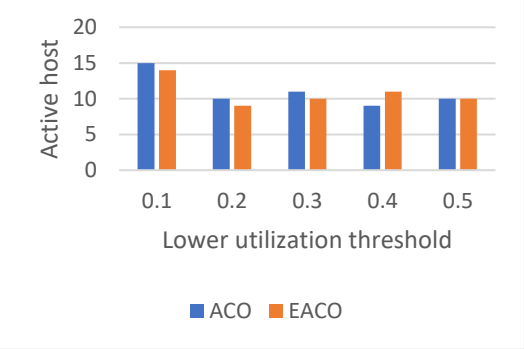

Fig 3 Active Host Vs Threshold

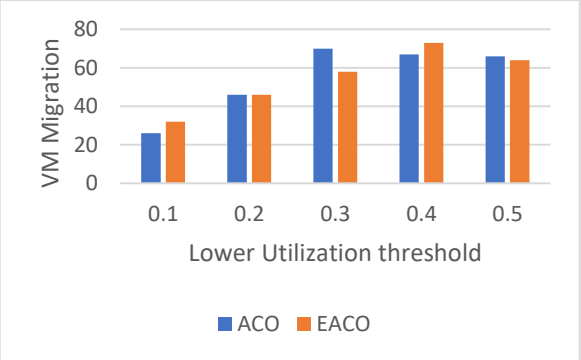

Fig 4 VM Migration Vs Threshold

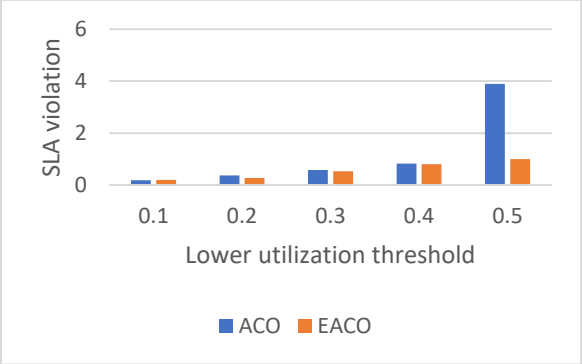

Fig 5 SLA violation Vs Threshold

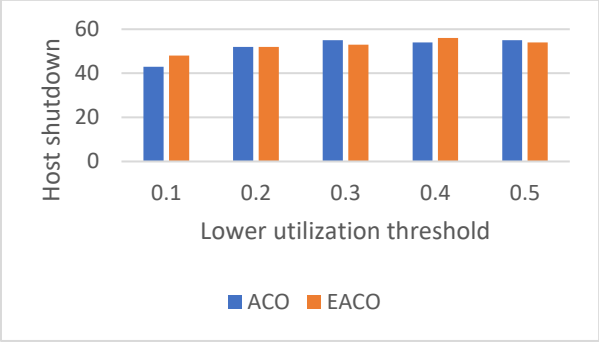

Fig 6 Shutdown host Vs Threshold

According to the X-axis and Y-axis, Figure 2 displays the threshold from 0.1 to 0.5 together with the matching energy values. Figure 3 displays the active host on the Y-axis together with the threshold, which ranges from 0.1 to 0.5. The threshold, which range from 0.1 to 0.5, are shown in Figures 4,5 and 6, together with the associated VM migration values, SLA violation and in active host on the Y-axis. The graph unequivocally shows that the EACA algorithm performs better than the fundamental ACO method. This result shows that, when compared to the standard ACO method, the suggested EACA algorithm performs better in terms of minimizing the energy, active host and optimizing the VM allocation. The graph unequivocally shows that the suggested EACA performs better than both the original ant colony algorithm approach. This result shows that EACA successfully optimizes VM allocation and achieves a perfect load balance throughout the entire system, resulting in a better-balanced system load.

5. Conclusion

In conclusion, it can be affirmed that our suggested methodologies outperform others across various scenarios, with the exception of the average SLA violation percentage. Notably, we observe a substantial reduction in energy consumption by up to 17.57%, accompanied by a decrease in the frequency of VM migrations. Furthermore, performance enhancements are evident across all approaches in diverse scenarios when compared to the dynamic threshold benchmark. We also present comparative graphs illustrating the application of dynamic load on VM. In this study, we employ the extended ant colony optimization, which aids in making informed decisions regarding VM migrations and selecting appropriate destination hosts. The evaluation of our proposed framework is conducted using the CloudSim simulator. The aforementioned outcomes underline the interconnected nature of energy consumption, VM migration frequency, average SLA violation, and performance decline.

Acknowledgements

The authors wish to extend their profound gratitude to the editor and reviewers for their invaluable assistance in enhancing and facilitating the publication of the manuscript.

Author contributions

Rudresh Shah contributed to design, implementation, and analysis of the results; Suresh Jain and Kailash Chandra Bandhu suggested for final manuscript preparation.

Conflicts of interest

The authors declare no conflicts of interest.

Reference

[1] P. M. Mell and T. Grance, “The NIST definition of cloud computing,” National Institute of Standards and Technology, 2011. doi: 10.6028/nist.sp.800-145.

[2] Y. Li and Z. Lan, “A Survey of Load Balancing in Grid Computing,” Heidelberg: springer Berlin, 2005.

[3] J. Luo, L. Rao, and X. Liu, “eco-IDC: Trade Delay for Energy Cost with Service Delay Guarantee for Internet Data Centers,” in 2012 IEEE International Conference on Cluster Computing, IEEE, Sep. 2012. doi: 10.1109/cluster.2012.23.

[4] K. W. Cameron, R. Ge, and X. Feng, “High-performance, power-aware distributed computing for scientific applications,” Computer (Long Beach Calif), vol. 38, no. 11, pp. 40–47, Nov. 2005, doi: 10.1109/mc.2005.380.

[5] Y. Ma, B. Gong, R. Sugihara, and R. Gupta, “Energy-efficient deadline scheduling for heterogeneous systems,” J Parallel Distrib Comput, vol. 72, no. 12, pp. 1725–1740, Dec. 2012, doi: 10.1016/j.jpdc.2012.07.006.

[6] X. Kong, C. Lin, Y. Jiang, W. Yan, and X. Chu, “Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction,” Journal of Network and Computer Applications, vol. 34, no. 4,

pp. 1068–1077, Jul. 2011, doi: 10.1016/j.jnca.2010.06.001.

[7] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers," Concurr Comput, vol. 24, no. 13, pp. 1397–1420, Oct. 2011, doi: 10.1002/cpe.1867.

[8] Clark Christopher, Keir Fraser, and S. Hand, "Live Migration of Virtual Machines," in 2nd Symposium on Networked Systems Design and Implementation, Boston Massachusetts, USA, May 2007.

[9] G. Khanna, K. Beaty, G. Kar, and A. Kochut, "Application Performance Management in Virtualized Server Environments," in 2006 IEEE/IFIP Network Operations and Management Symposium NOMS 2006, IEEE, 2006. doi: 10.1109/noms.2006.1687567.

[10] A. Beloglazov and R. Buyya, "Energy Efficient Resource Management in Virtualized Cloud Data Centers," in 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, IEEE, 2010. doi: 10.1109/ccgrid.2010.46.

[11] M. Forsman, A. Glad, L. Lundberg, and D. Ilie, "Algorithms for automated live migration of virtual machines," Journal of Systems and Software, vol. 101, pp. 110–126, Mar. 2015, doi: 10.1016/j.jss.2014.11.044.

[12] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," Future Generation Computer Systems, vol. 28, no. 5, pp. 755–768, May 2012, doi: 10.1016/j.future.2011.04.017.

[13] M. Andreolini, S. Casolari, M. Colajanni, and M. Messori, "Dynamic Load Management of Virtual Machines in Cloud Architectures," in Cloud Computing, Springer Berlin Heidelberg, 2010, pp. 201–214. doi: 10.1007/978-3-642-12636-9_14.

[14] S. B. Shaw and A. K. Singh, "Use of proactive and reactive hotspot detection technique to reduce the number of virtual machine migration and energy consumption in cloud data center," Computers &amp Electrical Engineering, vol. 47, pp. 241–254, Oct. 2015, doi: 10.1016/j.compeleceng.2015.07.020.

[15] M. Aldossary and K. Djemame, "Performance and Energy-based Cost Prediction of Virtual Machines Live Migration in Clouds," in Proceedings of the 8th International Conference on Cloud Computing and Services Science, SCITEPRESS - Science and Technology Publications, 2018. doi: 10.5220/0006682803840391.

[16] khan Shagufta and Sharma Niresh, "Effective Scheduling Algorithm for Load balancing using Ant Colony Optimization in Cloud Computing," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, no. 2, Feb. 2014.

[17] S Banerjee, I Mukherjee, and P Mahanti, "Cloud computing initiative using modified ant colony framework," in World Acad Sci Eng Technol, 2009, pp. 221–224.

[18] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud Task Scheduling Based on Load Balancing Ant Colony Optimization," in 2011 Sixth Annual Chinagrid Conference, IEEE, Aug. 2011. doi: 10.1109/chinagrid.2011.17.

[19] M. Sohani and Dr. S. C. Jain, "Threshold based VM Placement Technique for Load Balanced Resource Provisioning using Priority Scheme in Cloud Computing," International journal of Computer Networks &amp Communications, vol. 13, no. 5, pp. 1–18, Sep. 2021, doi: 10.5121/ijcnc.2021.13501.

[20] R. Mishra, "Ant colony Optimization: A Solution of Load balancing in Cloud," International journal of Web & Semantic Technology, vol. 3, no. 2, pp. 33–50, Apr. 2012, doi: 10.5121/ijwest.2012.3203.

[21] Joshi N. A., "Dynamic Load Balancing In Cloud Computing Environments," International Journal of Advanced Research in Engineering and Technology, vol. 5, no. 10, pp. 201–205, Oct. 2014.

[22] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Softw Pract Exp, vol. 41, no. 1, pp. 23–50, Aug. 2010, doi: 10.1002/spe.995.