https://doi.org/10.48048/tis.2022.4502

A Recursive Ant Colony Optimization Algorithm for Energy Consumption in Cloud Computing

Usha Kirana Sonangeri Pushpavathi[*] and Demian Antony D'Mello

Department of Computer Science and Engineering, Canara College of Engineering, Visvesvaraya Technological University, Mangaluru, India

([*]Corresponding author's e-mail: usha14.nayak@gmail.com)

Received: 25 December 2020, Revised: 6 May 2021, Accepted: 13 May 2021

Abstract

The Cloud Computing (CC) has vast amount of data centers that consists of many computing nodes and consumes a huge amount of electrical energy. Hence the researchers found that the high service-level agreements (SLAs) violations and Energy Consumption (EC) are the major challenging issues in CC. The various traditional approaches reduced the EC, but ignored the SLA violation during the selection of Virtual Machine (VM) from overloaded hosts. In order to effectively deal with these issues, this paper proposed the Recursive Ant Colony Optimization (RACO) in 2 different workloads. The main aim of the RACO is to minimize the high EC and SLA violations by the movement of best ant, which is random and identified the distance between their movements. The algorithm consists of 3 major steps includes tracking and updating the pheromone and finally city selection. The proposed method simulated on Cloud Sim to validate the efficiency and stability of the proposed RACO and their performance compared to that of other existing techniques. The results showed that RACO reduced the EC by 40 - 42 % (approx.) which is less than the traditional ACO algorithm in Planet Lab data.

Keywords: Ant colony optimization, Cloud computing, Cloud Sim, Energy consumption, Service level agreements, Virtual machines

Introduction

Nowadays, CC uses the Internet for connecting vast amount of storage as well as computing resources and it is also developed by fusing the parallel processing, grid and distributed computing. Moreover, the end-users obtained the cheap, dynamically scalable computing and storage resources services by using the CC application [1,2]. According to the actual usage of resources, the users can pay for the services, because on-demand access to those services is offered by CC. Therefore, a large-scale data centers are required to fulfill the demand of resources. The essential services to the cloud users are provided by requiring a huge amount of electric power in the data centers, but it leads to increase the computation as well as operational costs [3,4]. The energy can be converted by using the VM consolidation in cloud data centers, which is considered as an efficient way. However, the performance may be degraded due to VMs' aggressive consolidation and response time or even failures are increased. This is because of underlying physical resources are shared by VMs and the requirement of an unexpected resources are encountered by applications [5-7]. The different computing tasks or reservations of resources must be submitted to the cloud providers by the cloud users. The revenue is generated by major 3 services of cloud providers namely software as a service (SaaS), infrastructure as a service (IaaS) and platform as a service (PaaS).

The customers should sign the SLA with cloud providers, before obtaining any cloud services, so it is important to provide better Quality of Service (QoS) to the end users by providers [8,9]. The current network traffic is not considered by the providers as well as users, which may leads to performance degradation. The SLA violations are not really reduced under a different kind of workloads; hence it is important to make a tradeoff between performance and energy to reduce the EC and meet the SLA violations [10,11]. In this research study, a new enhanced ACO called Recursive - ACO is implemented to reduce the SLA violation and minimize the high EC. The social behavior of ant is simulated by the ACO algorithm, where the ant path from the nest to food sources are optimized. According to the pheromone laid on the path, the shortest path is identified, because the ant's movement is random. The recursion of ACO is extended by applying an additional term 'depth' using the proposed RACO method. The VM allocations

are effectively carried out by using the Inter Quartile Range (IQR). The simulations are conducted to test the proposed RACO's performance in terms of EC, VM migrations and number of host shutdown.

The organization of the research paper consists of: Section 2 discusses the existing techniques with its limitations that are developed in cc for sla violations. the system model with the problem formulation is described in section 3. the explanation of proposed methodology is presented in section 4. the validation of proposed raco with traditional approaches is discussed in section 5. finally, the conclusion of the research study with future work is illustrated in section 6. in this paper resources are termed as assets.

Literature review

In this section, a discussion of existing techniques [12-16] that are developed to reduce the SLA violation and high EC in Cloud Computing. In addition, the key benefits of existing methods with its limitations are presented.

Li *et al.* [12] implemented a host overloading/under-loading detection algorithm according to simple linear regression model in order to minimize the power consumption and SLA violation. While comparing with other native linear regression model, the errors were directly or indirectly added to the prediction, because the squint toward over-prediction was amended by the developed method. The errors were calculated by implementing the 8 methods. The 2 workloads namely random and planet lab were used to extend the Cloud Sim simulator for testing the performance of developed detection algorithm.

Yadav *et al.* [13] minimized the SLA violation and consumption of energy by designing 3 adaptive models such as Maximizing Correlation Percentage (MCP), Bandwidth - concerned policy (Bw) and Gradient descent-based regression (Gdr). According to robust regression model, adaptive energy-aware algorithms such as Gdr and MCP were used for finding the over burned host. The key strength is, Virtual Machine migration, average SLA violation, the count of closed hosts and consumption of energy. The obtained simulation experimental results stated that the Gdr improved the EC, while comparing with the MCP. The drawback is that the EC of the data center and the count of closed hosts increase because of high reactivation of hosts.

Zhou *et al.* [14] minimized the rate of SLA violations and increased the energy efficiency by developing 2 novel adaptive energy-aware algorithms in cloud datacenters. In this study, the 2 VM selections methods were considered from the overloaded hosts for migrating the VM. To make an effective decision, the developed method used both memory utilization as well as CPU. The Planet Lab was used to test the efficacy of the adaptive methods with traditional existing techniques. While maintaining the low SLA violations, the method minimized the EC in the datacenters, but the operation cost of energy - aware algorithms was increased.

The 2-level management model implemented [15] for heterogeneous cloud environment for optimizing the various heuristic strategies. The historical data of the host was analyzed by predicting the future state of the host using empirical forecast host detection algorithm (EFA). The migration priority was calculated by implementing algorithm called Weighted Priority for suitable Virtual Machine selection. The simulations results stated that the EFA - WPA achieved better performance, while comparing with traditional overloaded host detection algorithms under various workloads. The network communication overhead between VMs was not considered in this developed method as it leads to high traffic between Virtual Machine migrations which is a major drawback of [15].

Hussain *et al.* [16] utilized the resources efficiently and minimized the execution time by designing the cost-efficient SLA-based Resource-Aware Load Balancing Algorithm (SLA-RALBA). Among the execution time, cost and resource utilization, an improved balance was achieved by the SLA-RALBA. The HCSP and GoCJ datasets were used in the experimental analysis for the validation of developed algorithm. The strengths are Average Resource Utilization Ratio (ARUR), make span and expected gain were used in the SLA-RALBA. However, the drawback is, the developed method consumed high energy and increased the workload due to vast number of tasks.

Aliyu *et al.* [17] treated the problems of slow convergence, unbalanced load, speed and low utilization of Virtual Machine resources by using local optimization technique. Konjaang *et al.* [18] proposed a methodology to reduce the length of the task by splitting the task into sub-tasks by employing the MOWOS algorithm. In this way it reduces the make span, cost and resource utilization.

Therefore, in order to deal with the issues of existing techniques, the research study is focused on the optimization algorithm called RACO to minimize the EC and SLA violations, which is explained in subsequent sections with required system model.

System model

In a distributed computing area, more than 1 heterogeneous Virtual Machine can be introduced on a single host. The virtualization innovation gives managerial benefits to Virtual Machine clients inside the view of guest who is working in the framework in that they can redo their run-time assets as per particular concerns [17]. The selected Virtual Machines can run various sorts of the service application at the same time. Each Virtual Machine and host is described by parameters, for example, CPU figuring power characterized in million instructions for every second (MIPS), auxiliary stockpiling gave by a system appended capacity or capacity territory arrange, essential stockpiling (RAM), and system transmission capacity [18]. In this design, the primary goal of effective EC with the advantages of Virtual Machines can be accomplished by combining the calculation load into few servers while setting inert servers to a vitality sparing mode [19]. The fundamental issue of EC is to identify the over-burden host to locate it from dynamic server farm to coordinate the present workload. This issue becomes testing when the dynamic idea of server farm are perceived, in that Virtual Machines can demand assets whenever and discharge the assets when their work is finished. In this way, different heterogeneous Virtual Machines demand assets and leave whenever they are done with its process. This distributed computing engineering can be viewed as a profoundly powerful framework.

The Recursive based calculations are particularly valuable in the dynamic idea of cloud designs. This situation permits the analysts to perceive the exhibition of this mind boggling framework and is valuable in structuring server farm asset the executives approaches. **Figure 1** shows the architecture of proposed CC with 3 layers.

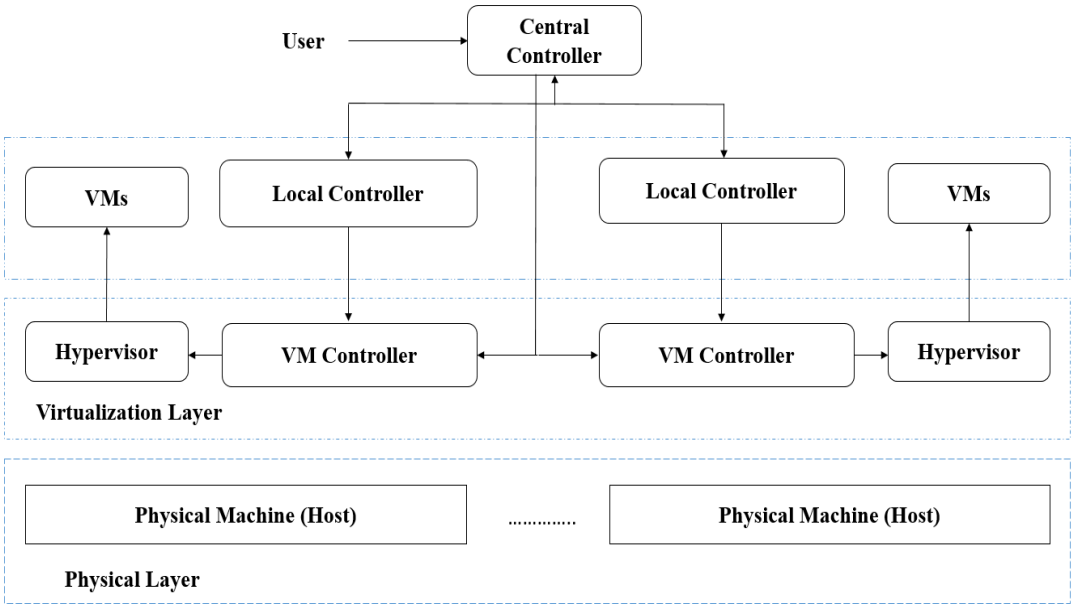

Figure 1 System model.

In the proposed framework, the bottom layer of this design contains number of physical servers (has) that comprise of an enormous size of power computing resources and capacity assets. Upcoming layer has virtualization facility, their clients can demand their assets to run the heterogeneous application as Virtual Machines and run their undertakings with various Service Level Agreements. In the proposed model, the 3 main players are centralized, neighborhood and Virtual Machine controllers. A nearby controller dwells in each host as a different Virtual Machine and screens the status of the Virtual Machine and CPU usage. The nearby controller ought to have the option to choose when VM ought to be moved from the host. The focal controller dwells in the single ace host and accumulates all data from the nearby controller to keep up the general usage of assets. The focal controller settles on the ideal area of the Virtual Machine. At last, the Virtual Machine controller dwells in the hypervisor and it is helpful in resizing the Virtual Machine and changing the force condition of the host. The step-by-step work process of these 3 controllers is portrayed as follows:

1) The nearby controller of each host keeps on observing the CPU usage of the host and sets it into 3 explicit fields, in particular, to over-burden host underloaded host and typical host.

2) The centralized controllers keep on social affair data from the nearby controller about the general perspective on the asset usage of the hosts and structure the best ideal arrangement for Virtual Machine position through Recursive-ACO as RACO. It gives the order to the Virtual Machine controller for union.

3) The Virtual Machine controller is helpful in relocating and resizing the VMs.

Problem formulation

The main objective of the proposed RACO is to reduce the EC and SLA violation for cloud providers under heterogeneous cloud environment. Initially, the problem formulation of EC is defined in the following sub-section.

Definition of energy consumption

The EC of computing nodes in cloud data centers is mainly determined by the disk storage, CPU, memory and network interfaces. The paper contains, the major aim is to study the EC generated by the central processing unit. Generally linear model is used to estimate the EC of host, where there is a linear correlation between EC and CPU utilization. Therefore, the definition of the EC of the i^{th} server at time t in the proposed method is given as Eq. (1);

$$EC_i(t) = \begin{cases} \rho \times EC_i^{full} & if\ the\ H_i\ is\ idle, \\ EC_i^{idle} + (1-\rho) \times EC_i^{full} \times U_i(t) & if\ the\ H_i\ is\ busy. \end{cases} \quad (1)$$

The point where the server is latent, ρ is a static coefficient showing to the imperativeness extent of the inactive processor (i.e., 70 %). The EC_i^{full} represents the EC of the physical present node i under fully loaded. Since the CPU usage progressively changes as indicated by the remaining task at hand, the CPU use is an element of the time t as a free factor, which is meant as $U(t)$. The energy utilization of server H_i throughout the process can be represented as in Eq. (2);

$$EC_i = \int_{t_0}^{t_1} EC_i(t)\, dt \quad (2)$$

Then, the total EC of a cloud data center with n nodes is given in Eq. (3);

$$EC = \sum_{i=1}^{n} x_i EC_i \quad (3)$$

where, $x_i = \begin{cases} 0 & if\ the\ H_i\ is\ shutdown \\ 1 & other \end{cases}$.

Definitions of SLA violation metrics

In a cloud server farm, the SLAs are formal portrayal of the QoS that cloud service providers or suppliers give to cloud users or clients. A SLA infringement happens when clients present an abundance request to the server farm. In the event that the host system oversubscribes, its figuring execution will be extraordinarily decreased. So, the Virtual Machine movement process devours extra assets. The level of SLA infringement legitimately mirrors the accessibility and heartiness of the framework. This examination study selects different measurements characterized in to quantify the level of SLA infringement.

1) SLATPAH: the level of time every dynamic host disregarded the Service Level Agreement, which is given in Eq. (4);

$$SLATPAH = \frac{1}{n} \sum_{i=1}^{n} \frac{T_i^f}{T_i^a} \quad (4)$$

where, the number of servers is described as n; Total time that host i as T_i^a, which is active; the total time that host system i experiences full load as T_i^f.

2) SLAPDM: the host performance degradation caused by VM migration that is expressed as Eq. (5);

$$SLAPDM = \frac{1}{m}\sum_{j=1}^{m}\frac{c_j^d}{c_j^r} = \frac{1}{m}\sum_{j=1}^{m}\frac{0.1\times c_j}{c_j^r} \quad (5)$$

where, the number of Virtual Machines in the data center is illustrated as m; total required CPU capacity (in MIPS) during the life cycle of VM j as C_j^r; a ruledout performance due to VM j migration as C_j^d. It is generally believed that the Virtual Machine migration will lose 10 % of the CPU computing power. The CPU power of Virtual Machine j is represented as C_j. Therefore, the overall Service Level Agreement violation is defined as the product of Eqs. (4) and (5), i.e.,

$$SLAV = SLAPAH \times SLAPDM \quad (6)$$

Therefore, these 2 problems must be minimized by using the proposed method called RACO, which is discussed in the below section.

Proposed methodology

In this section, a Recursive-ACO is designed to reduce the SLA violation and maximize the resource utilization. The following **Figure 2** shows the working procedure of Recursive-ACO.

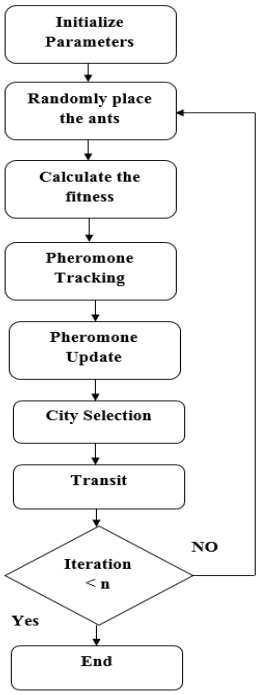

Figure 2 Flow of proposed method: Recursive-ACO.

The ACO is a populace based calculation which recreates the social conduct of ants and has been proposed as stochastic and meta-heuristic calculation to take care of different combinatorial issues. Here, an expansion to the ACO calculation named as Recursive-ACO is proposed, an altered type of ACO. The fundamental component of ACO is the utilization of the pheromones (comparable to the pheromones laid by ants) to create better arrangement as emphases progress. All the subterranean insect settlement calculations proposed till date share some normal highlights. The space of the issue should be appropriately demonstrated into urban areas or ways where the ants dwell during their inquiry of a decent arrangement. The exactness of the conclusive outcome is subject to the surrounded area. The fitness function is

calculated, once all the ants are randomly placed. From the general ACO model, it is discovered that if the quantity of models are less or the quantity of cycles are less, at that point no such arrangement might be acquired whose pheromone trail force is extremely high. Or maybe a few urban areas with confinement of the ants can be seen. Likewise, the mistake might be huge and unsatisfactory. To reduce this error, Recursive-ACO algorithm is proposed.

Pheromone tracking

The quantity of qualities that can be created inside a given range can be endless. Be that as it may, it is unrealistic to state that the pheromone power for every such value can be refreshed. The presentation of urban areas discretizes this range and makes it conceivable to monitor just a limited no of urban communities (values). The ants move between the urban areas driven by the pheromone force at these urban communities.

Pheromone updating

The quantity of urban areas is substantially more than the quantity of ants. It is beyond the realm of imagination to expect to refresh the pheromone of every single city. Or maybe the pheromone is refreshed distinctly for those urban communities where the ants move. This implies the pheromone dissipation isn't considered for purpose of decreasing the unpredictability.

City choice after every profundity

The ants are restricted at urban communities close to the nearby maxima. In the event of ACO, there might be a few focuses which fulfill the neighborhood maxima test; however they may not be real maxima states. These can be excluded by the Recursive-ACO strategy with expanded number of profundities. Consequently, rough dataset can be handily taken care of utilizing Recursive-ACO. Be that as it may, there must be heuristics to diminish their tally before continuing into the following profundity else countless urban communities going into the following profundity may build the time and multifaceted nature. One of the ways is to change the technique for checking for neighborhood maxima to check for close by urban communities a good way, out of 2 or might be more prominent instead of simply checking the following neighbors.

Consider n as cities and m as ants, the number of cities is fully connected with the edges E^n. The proposed optimization method is explained in step-by-step as follows:

1) Initialization: Place the m ants in the n cities randomly, and pheromone value is assigned as small positive variable.

2) Path construction: Every ant chooses the upcoming city to visit, based on the transition probability as defined as in Eq. (7).

$$p(i,j) = \begin{cases} \frac{[\tau(i,j)]^{\alpha}[\eta(i,j)]^{\beta}}{\sum_{u \in J}[\tau(i,u)]^{\alpha}[\eta(i,u)]^{\beta}}, & if\ j \in J \\ 0, & otherwise \end{cases} \quad (7)$$

where ant k is in the city i and the next city is j, τ is the amount of pheromone on the edge (i,j), the heuristic edge value is η_{ij}, the 2 weighting factors that controls the significance between the pheromone and heuristic data α and β, and set of cities refers to J is not visited.

3) If all the cities are completely visited by all the ants, then go to step 4. If fails, go to step 2.

4) Updating the Pheromone by using the Eq. (8);

$$\tau(i,j) \leftarrow (1-\rho).\tau(i,j) + \rho.\Delta\tau(i,j) \quad (8)$$

where $\rho \in (0,1)$ are the evaporation rate and the value $\Delta\tau\ (i,j)$ is related to the fitness function value.

5) Set the stop-point criteria as distance between the best ants' route as $X = (x_1, x_2, \dots, x_r)$ and weak ants' route as $Y = (y_1, y_2, \dots, y_s)$, which is described in the Eq. (9);

$$d = \sum_{i=1}^{n} |X_i - Y_i| \quad (9)$$

where, the number of variables as n, the values of the i^{th} variable as $X_i\ and\ Y_i$ at points $X\ and\ Y$, respectively.

6) When final criteria are reached, it stops the process, otherwise randomly place the ants and follow step 2, and in this step the good solution is obtained [20].

Therefore, the proposed RACO will achieve the optimization problem effectively, i.e. it highly minimized the energy consumption as well as SLA violation. The validation of the proposed method with existing techniques is presented in the next section.

Complexity analysis

The issues of the domain have to be identified and modeled into routes. The algorithm has following steps: Routes through which ants are moving repetitively and these routes are connecting the cities. The best routes are given by pheromone trails and heuristic function for updating the global pheromone. Probability of a route is chosen by an ant when moving between routes.

In the algorithm stated above, every ant has to be updated for each repetition. Numbers of ants are denoted as $\prod_{b=1}^{m} S_b$, the complexity can be stated as;

$$\tau - t * (\prod_{b=1}^{m} S_b)$$

where, S_b is the number of the ants and D_b is the set of models which is adopted by the algorithm. General ACO is different from the single depth RACO algorithm in many ways and that in general ACO local maxima check can be carried in each city, whereas in RACO only the solutions are considered. Therefore, time complexity of ACO will be much more comparatively to RACO;

$$\tau = O (\prod_{b=1}^{m} D_b)$$

Hence RACO consumes lesser time for a solution of the problem comparing to the general ACO algorithm. When accuracy is concern RACO has an upper hand to general ACO algorithm.

Results and discussion

In this section, the exploratory arrangement and trial results are depicted. Cloud Sim distributed computing reenactment stage is considered for calculation execution testing, which gives the accompanying highlights:

1) A supporting tool for modeling and for huge scope simulation of Cloud Computing framework.

2) An independent stage supporting server farms, administration operators, scheduling and distribution techniques.

Consequently, the Cloud Sim is truly reasonable for distributed computing reproduction tests.

Experimental setup

The proposed technique, reenact the calculations under 2 sorts of outstanding workloads as to be specific Planet Lab and Random workloads. Here, The Planet Lab information is given as a piece of the Co Mon venture, which is a checking framework for Planet Lab. From this venture, CPU uses information which is obtained from in excess of a thousand Virtual Machines of servers at every 5-min interims, and these servers are situated in excess of 500 areas around the globe. The information is to put away in 10 distinct documents. The proposed RACO chose 2 days from the outstanding task at hand following. Through reenactment, each Virtual Machine haphazardly circulates outstanding burden following from one Virtual Machine on the comparing date. In the Random outstanding task at hand, each Virtual Machine runs an application with the variable remaining task at hand, which is displayed to create the use of CPU as indicated by a consistently dispersed irregular variable. As a matter of fact, at whatever point an irregular capacity creates a lot of remaining task at hand naturally as a cloudlet, called Random outstanding task at hand.

According to Planet Lab, 2 types of heterogeneous physical host instances are described in this proposed method. The one host instance type is HP ProLiant ML 110 G 4, which has 4 GB RAM, 2 Processing elements and 1860 MIPS. The other type of host instance is HP ProLiant ML110 G5, which also have 2 processing elements, 4 GB RAM and 2660 MIPS. According to Amazon EC 2, 4 heterogeneous VM types are defined in **Table 1**.

Table 1 Types of VM instances.

VM instance Types	VM_MIPS	VM_RAM
Small Instance	1000	1.70 GB
Micro Instance	500	0.61 GB
High - CPU Medium Instance	2500	0.85 GB
Extra Large Instance	2000	3.75 GB

Existing techniques

The proposed RACO method is compared with EFA - WPS [15] as EfWp by means of EC, SLA violations, total number of VM migrations and number of host shutdowns. In addition to existing EfWp, some important existing algorithms are described in the below section.

1) MeMs [15]: This method creates an upper CPU usage by limit utilizing M-GAUGE regression to identify over-burden hosts and later joins the MuMs Virtual Machine determination plan to improve Virtual Machine combination.

2) LrMmt [15]: The fundamental thought of LrMmt is to fitting the basic models to confined subsets of information to develop a curve that approximates the initial information with considering the movement time of the Virtual Machine.

3) MadMu [15]: The principle thought of MadMu is to modify the estimation of the upper use limit contingent upon the quality of the deviation of the CPU use. In the Virtual Machine determination stage, the calculation chooses the Virtual Machine with the most reduced CPU use for relocation. After numerous emphases, the host use is beneath the upper limit.

4) IqrMc [15]: The IQR which is known as the mid-spread or center 50, is a proportion of measurable scattering, being equivalent to the contrast between the third and first quartiles. The IQR is a strong measurement, having a breakdown purpose of 25 % and is in this manner frequently liked to the all-out range. The IqrMc chooses those Virtual Machines to be moved that have the most noteworthy relationship of the CPU usage with different Virtual Machines.

Parameter metrics

In the proposed method 4 major parameters are considered, namely; EC, Service level agreement violations, count of shutdown host and count of virtual machine migrations. The EC and SLA violations are already explained in section 3. The remaining parameters metrics are described as follows:

Total number of host shutdowns

The extreme number of host deactivation and reactivation causes additional costs for conjuring the hosts back because of unexpected expanding the interest of assets. The proposed approaches switch underloaded or perfect host to vitality sparing mode to limit the EC. Be that as it may, inordinate shutdowns cause additional endeavors and deferral to repudiate them when assets request emerges.

Number of VM migration

The migration of live VMs is a costly activity as far as SLA, there are few factor legitimately sway on it: (a) the system traffic between the source host and destination host; (b) Service shutdown time over the VM relocation; and (c) Time of VM total migration. The 10 % of CPU usage of host vitality utilization is adding to SLA during all VM relocations in the hosts.

Analysis of proposed RACO in random data

In this section, initially, the experiments are conducted on random data to test the efficiency of RACO with ideal ACO and other existing techniques [15]. **Table 2** shows the validated results of proposed RACO by means of 4 parameters. Initially, the EC for all existing techniques with proposed RACO are graphically expressed in **Figure 3**.

Table 2 Performance analysis of proposed raco for random data.

Methods	Energy consumption (kWH)	SLAV($\times 10^{-10}$)	No. of host shutdowns	No. of VM migration
MadMu	90	0.5	6400	11,000
IqrMc	89	0.8	5500	10,000
LrMmt	69	1.0	5000	7000
MeMs	62	2.8	2800	8000
EfWp	60	0.5	2000	5000
Ideal-ACO	49	0.2	1622	4800
Proposed RACO	**48**	**0.1**	**1119**	**4500**

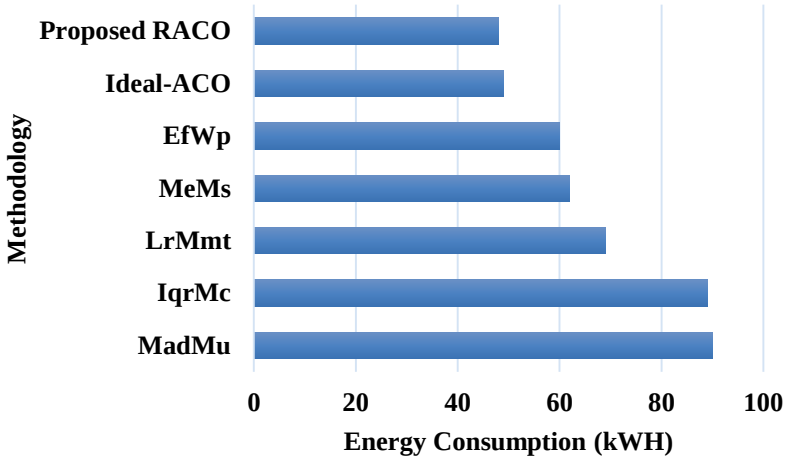

Figure 3 Performance of proposed method in terms of EC.

From **Table 2**, the existing techniques: MadMu and IqrMc consumed high energy (i.e. 90 and 89 kWh) for the random data. In order to address these high EC of existing techniques, the LrMmt, EfWp and MeMs are developed; however, it also reduced the energy in only limited amount. While comparing with MeMs algorithm, the EfWp achieved better performance, because the interaction effects and nonlinear causality are ignored by the MeMs algorithm. But, when compared with ideal ACO, the EfWp increased the EC, i.e. it consumed 60 kWh, where ideal ACO consumed only 49 kWh. The reason is that the ideal ACO considered the effect of recent energy utilization history records on host future status. But, the stopping criteria are not defined in the ideal ACO, that leads to consumed more energy. To avoid these issues, the RACO is proposed and set the distance between ants and weak ants. Therefore, the proposed RACO consumed only 48 kWh on random data. **Figure 4** shows the graphical representation of SLA violations.

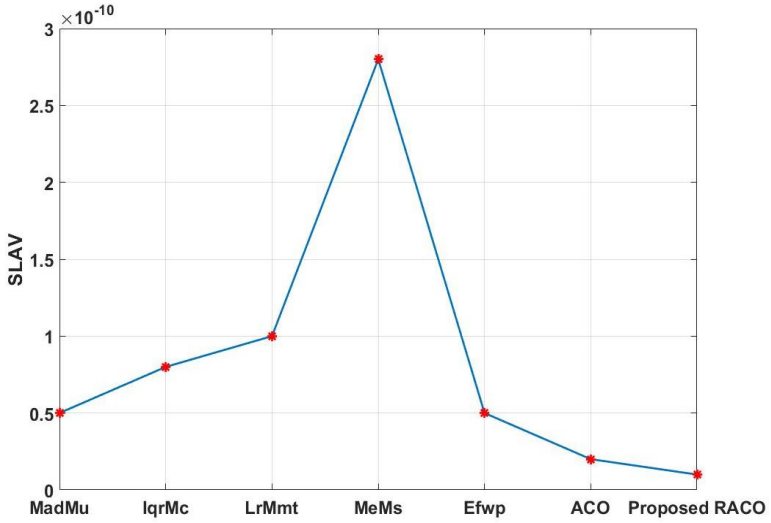

Figure 4 Analysis of SLA violation for proposed RACO.

When compared with every technique, MeMs have increased SLA violations, because the LrMmt focused only on minimizing the EC with the improvement of QoS as an auxiliary effect. But, the EfWp and MadMu reduced those SLA violations with other techniques includes IqrMc and LrMmt. While the optimization techniques are used, the SLA violations are further reduced, because it uses the stopping criteria functions to minimize the SLA. Therefore, the proposed RACO achieved less SLA violations, when compared with other existing and ideal techniques. **Figure 5** presents the performance of RACO in terms of number of host shutdowns.

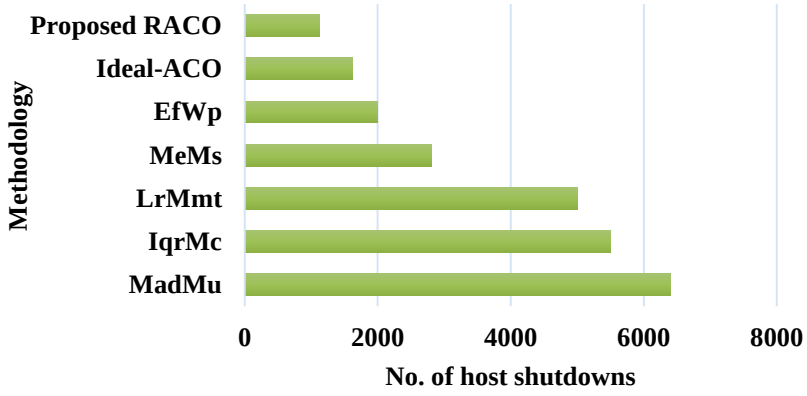

Figure 5 Graphical representation of no. of host shutdown for proposed method.

From the **Table 2**, it is clearly stated that the proposed RACO achieved better performance for reducing the number of host shutdowns, which leads to improve the QoS. When compared with MadMu and IqrMc, the proposed RACO minimized the host shutdown up to 45 - 55 %. The existing techniques MeMs and EfWp has the total number of shutdown are 2800 and 2000 for random workload. When the compared with the ideal ACO (i.e. 1622), the RACO has the lowest shutdown (i.e. 1119), which proves that the proposed method achieved better performance. This is because, the host status at the next moment are predicted by the RACO, which is based on the recent host utilization rate. Therefore, the proposed method accurately captured the host load and the restart of the shutdown host is avoided. Finally, the number of VM migrations is shown in **Figure 6**.

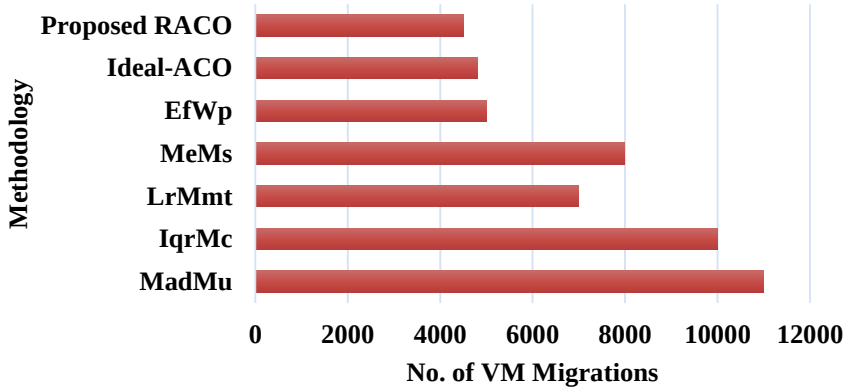

Figure 6 Performance analysis of number of VM migrations for proposed method.

The Virtual Migration migration of proposed RACO is highly reduced (i.e. 4500) than the VM migration of MadMu and IqrMc (i.e. 11,000 and 10,000). The existing techniques such as LrMmt and MeMs have the total number of Virtual Migration migrations as 7000 and 8000. The EfWp has less VM migrations, while comparing with other existing algorithms, which leads to have higher QoS. But, when the heuristic optimization algorithms are used, the VM migrations are highly reduced, for instance, the ideal ACO achieved only 4800 VM migrations. The ideal ACO are used with simple mechanisms, but the proposed RACO works effectively by finding the distance of best ants and weak ants. Therefore, the proposed method achieved only 4500 VM migrations than other techniques. In the next sub-section, the validation of proposed method on Planet Lab data is explained.

Analysis of proposed RACO in Planet Lab data

The experiments are conducted on Planet Lab data to verify the performance of proposed RACO with various existing and ideal ACO algorithm. **Table 3** shows the validated results of proposed method in terms of EC, number of hosts shut down, SLA violations and total VM migrations and **Figure 7** shows the EC of various techniques.

Table 3 Performance analysis of proposed RACO algorithm for Planet Lab data.

Methods	Energy consumption (kWh)	SLAV ($\times 10^{-10}$)	No. of host shutdowns	No. of VM migration
MadMu	170	2.6	5500	3000
IqrMc	200	2.8	6200	2500
LrMmt	160	3.0	5000	3800
MeMs	125	8.2	2800	1800
EfWp	120	1.8	2000	900
Ideal-ACO	85	1.5	1800	800
Proposed RACO	**49**	**0.9**	**1600**	**600**

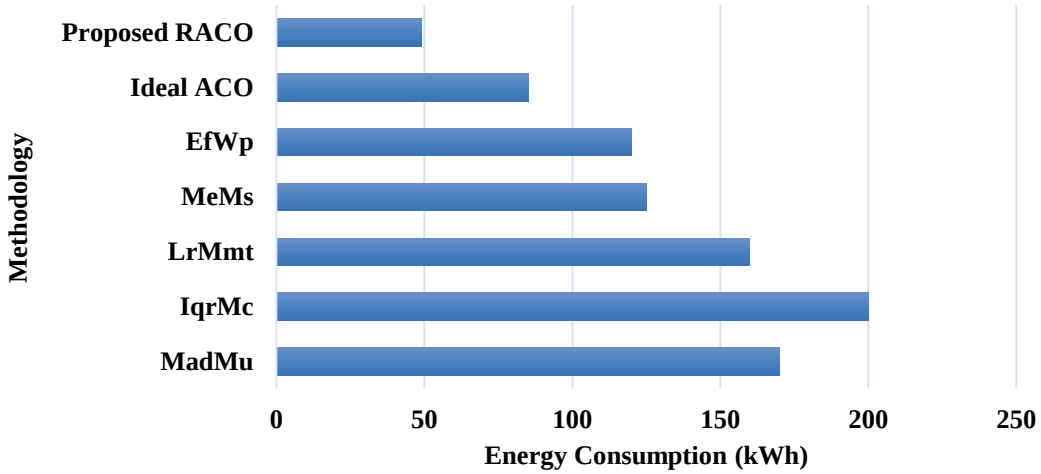

Figure 7 Performance of RACO for Planet Lab data in terms of EC.

From the **Table 3,** it is clearly stated that the proposed RACO consumed very less energy than ideal ACO and other existing algorithms. For instance, the EfWp consumed 120 kWh energy and ideal ACO consumed 85 kWh energy, but the proposed RACO achieved only 49 kWh energy. When compared with other existing techniques, the IqrMc consumed very high energy for the Planet Lab data (i.e. 200 kWh). The existing MadMu and LrMmt consumed 170 and 160 kWh energy, where the MeMs consumed only 125 kWh energy. The proposed RACO consumed 48 kWh energy for random data and 49 kWh energy for Planet Lab data. This is because the virtual machine selection method in RACO is based on the recent virtual machine concerns and minimum migration time. The effect factor is boosted and weighted to accomplish precise VM determination. Thus, RACO is well fitted energy utilization algorithm because of its straightforward energy calculation count with different methodologies. The validation results of SLA violation are graphically represented in **Figure 8**.

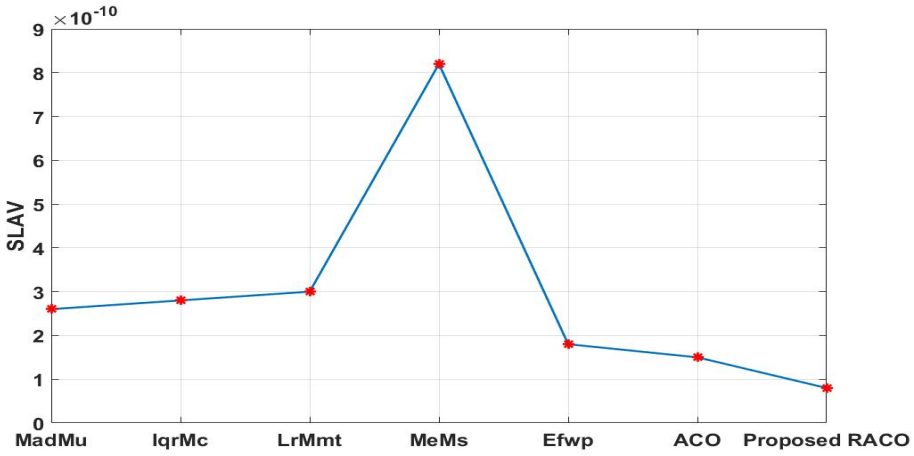

Figure 8 SLA violation of proposed method for Planet Lab data.

When compared with random data, the performance of both existing and proposed method obtained high SLA violations. Among the other existing techniques, the MeMs achieved high SLA violations, where LrMmt achieved 3.0×10^{-10} SLA. The proposed method nearly reduced the SLA violations upto 25 - 30 % than the EfWp, MadMu and IqrMc techniques. Here, the proposed RACO has higher SLA violations in Planet Lab data than random data. The next analysis includes number of host shutdowns, which is depicted in **Figure 9**.

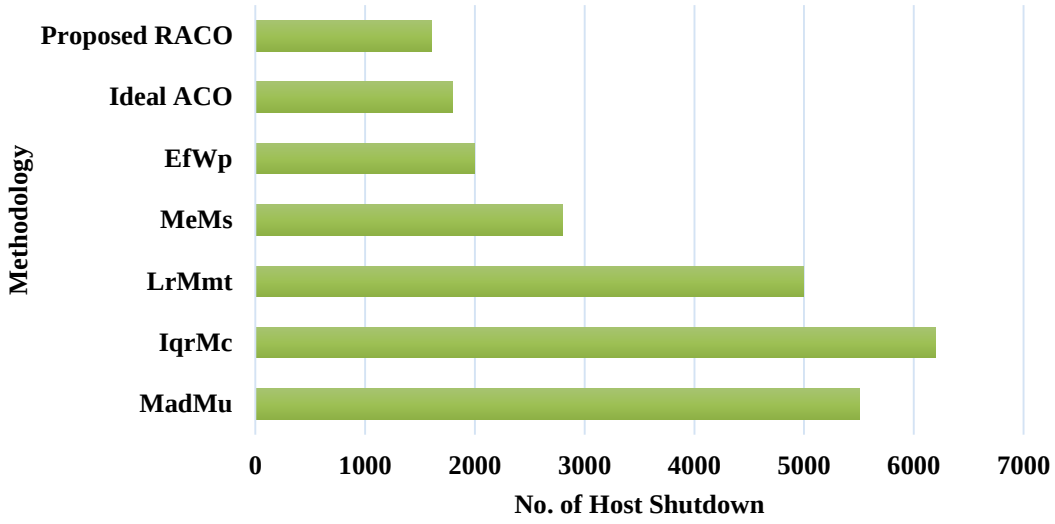

Figure 9 Analysis of proposed method for planet lab data by means of host shutdown.

While comparing the performance of proposed RACO for random data, the developed method achieved higher host shutdowns for Planet Lab data. For instance, the RACO achieved 1600 host shutdown for Planet Lab data and it achieved only 1119 host shutdown for random data. In addition, the other existing techniques are also having higher host shutdown for Planet Lab data, where IqrMc had the highest host shutdown (i.e. 6200) than other traditional and proposed RACO techniques. The existing techniques MadMu and LrMmt had 5500 host shutdowns and 5000 host shutdowns, where the MeMs and EfWp had only 2800 host shutdowns and 2000 host shutdowns. The ideal ACO have 1800 host shutdown, where the proposed RACO have only 1600 host shutdown. Finally, the graphical representation of number of VM migration is shown in **Figure 10**.

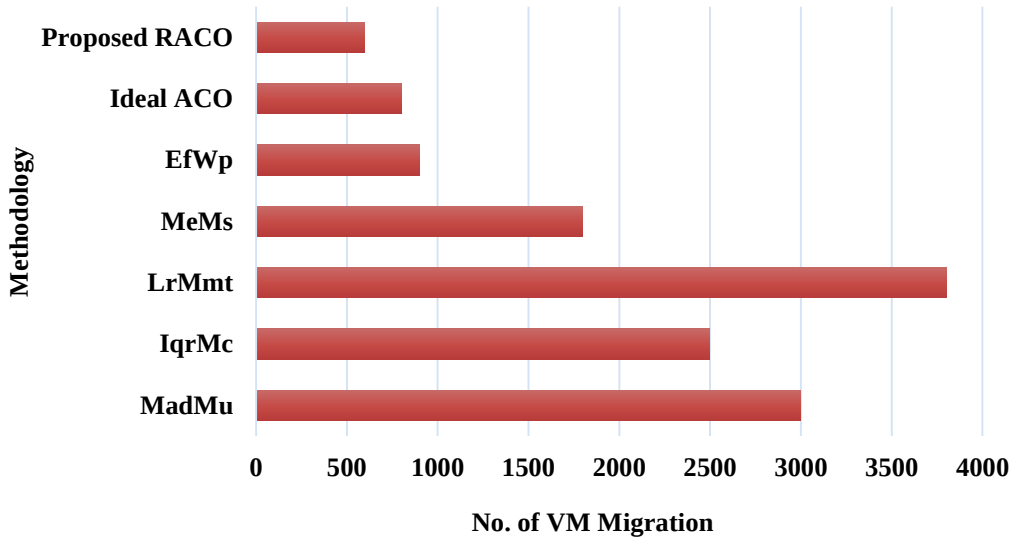

Figure 10 Total number of VM migrations for RACO method.

From the **Table 3** and **Figure 10**, it is clearly stated that the total number of VM migrations are reduced in the proposed RACO than ideal ACO algorithm. For instance, the ideal ACO had 800 VM migrations, where the proposed RACO has only 600 VM migrations. The MadMu and LrMmt had the higher VM migrations than other existing techniques (i.e. 3000 and 3800 VM migrations). However, it is highly minimized by EfWp (i.e. only 900 VM migrations) and IqrMc had 2500 VM migrations. When compared with random data, the proposed RACO highly minimized the VM migrations in the Planet Lab data. Therefore, the proposed RACO achieved better performance in reducing the EC and SLA violations than other traditional techniques. In order to improve the energy efficient and reduction of SLA violation, the further work is planned to optimal the proposed method by considering RAM.

Conclusions

In the data center, the optimizing between EC and QoS is a challenging situation for service providers and cloud users. The main objective of this paper is to seek the best compromise between EC and service quality. For this purpose, the RACO algorithm is designed, which has various advantages over other optimization techniques. To assign every tasks to the VM, RACO doesn't demand any initial values and it searches evenly in the given range as supplied by the user. When compared with ideal ACO, the time taken for allocating the tasks to VM by RACO is very less. The simulations are carried out to test the performance of RACO in Cloud Sim than other existing techniques. The results show that proposed RACO consumed only 48 and 49 kWh, where the ideal ACO consumed 49 and 85 kWh for random and Planet Lab data. But, the RACO has higher host shutdown (i.e.1600) in Planet Lab data, this is because it has various instance types with different MIPS. In future work, an improved optimization algorithm is developed to reduce the greater number of host shutdowns for Planet Lab data.

References

[1] SS Gill, L Chana, M Singh and R Buyya. CHOPPER: An intelligent QoS-aware autonomic resource management approach for cloud computing. *Cluster Comput*. 2018; **21**, 1203-41.
[2] Y Wang, X Tao, F Zhao, B Tian and AMVV Sai. SLA-aware resource scheduling algorithm for cloud storage. *EURASIP J. Wireless Comm. Network*. 2020; **2020**, 6.
[3] J Prassanna and N Venkataraman. Adaptive regressive holt-winters workload prediction and firefly optimized lottery scheduling for load balancing in cloud. *Wireless Network*. 2019; **27**, 5597-615.
[4] MH Malekloo, N Kara and ME Barachi. An energy efficient and SLA compliant approach for resource allocation and consolidation in cloud computing environments. *Sustain. Comput. Informat. Syst*. 2018; **17**, 9-24.
[5] M Ranjbari and JA Torkestani. A learnining automata-based algorithm for energy and SLA efficient consolidation of virtual machines in cloud data centers. *J. Parallel Distr. Comput*. 2018; **113**, 55-62.
[6] CD Martino, S Sarkar, R Ganesan, ZT Kalbarczyk and RK Iyer. Analysis and diagnosis of SLA violations in a production SAAS cloud. *IEEE Trans. Reliab*. 2017; **66**, 54-75.
[7] SK Panda and PK Jana. SLA-based task scheduling algorithms for heterogeneous multi-cloud environment. *J. Supercomput*. 2017; **73**, 2730-62.
[8] M Kumar and SC Sharma. PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing. *Neural Comput. Appl. 2019;* **32**, 12103-26.
[9] A Ramegowda, J Agarkhed and SR Patil. Adaptive task scheduling method in multi-tenant cloud computing. *Int. J. Inform. Tech*. 2019; **12**, 1093-102.
[10] D Komarasamy and V Muthuswamy. ScHeduling of jobs and adaptive resource provisioning (SHARP) approach in cloud computing. *Cluster Comput*. 2018; **21**, 163-76.
[11] S Mustafa, K Bilal, SUR Malik and SA Madani. SLA-aware energy efficient resource management for cloud environments. *IEEE Access* 2018; **6**, 15004-20.
[12] L Li, J Dong, D Zuo and J Wu. SLA-aware and energy-efficient VM consolidation in cloud data centers using robust linear regression prediction model. *IEEE Access* 2019; **7**, 9490-500.
[13] R Yadav, W Zhang, O Kaiwartya, PR Singh, IA Elgendy and YC Tian. Adaptive energy-aware algorithms for minimizing energy consumption and SLA violation in cloud computing. *IEEE Access* 2018; **6**, 55923-36.
[14] Z Zhou, J Abawajy, M Chowdhury, Z Hu, K Li, H Cheng and F Li. Minimizing SLA violation and power consumption in Cloud data centers using adaptive energy-aware algorithms. *Future Generat. Comput. Syst*. 2018; **86**, 836-50.
[15] X Zhou, K Li, C Liu and K Li. An experience-based scheme for energy-SLA balance in cloud data centers. *IEEE Access* 2019; **7**, 23500-13.

[16] A Hussain, M Aleem, MA Iqbal and MA Islam. SLA-RALBA: Cost-efficient and resource-aware load balancing algorithm for cloud computing. *J. Supercomput.* 2019; **75**, 6777-803.
[17] A Aliyu, AH Abdullah, O Kaiwartya, Y Cao, MJ Usman, S Kumar and RS Raw. Cloud computing in VANETs: Architecture, taxonomy, and challenges. *IETE Tech. Rev.* 2018; **35**, 523-47.
[18] IA Elgendy, WZ Zhang, CY Liu and CH Hsu. An efficient and secured framework for mobile cloud computing. *IEEE Trans. Cloud Comput*. 2018; **9**, 79-87.
[19] A Ahmed, AA Hanan, K Omprakash, MJ Usman and SOA Rahman. Mobile cloud computing energy-aware task offloading (MCC: ETO). *In*: Proceedings of the International Conference on Communication and Computing Systems, Gurgaon, India. 2016.
[20] E Papenhausen and K Mueller. Coding ants: Optimization of GPU code using ant colony optimization. *Comput. Lang. Syst. Struct*. 2018; **54**, 119-38.