DBSHD: Dynamic Burst-aware Source Host Detection in the Cloud

Somayeh Rahmani
Takestan Islamic Azad University
vahid khajehvand (vahidkhajehvand@gmail.com)
Qazvin University: Qazvin Islamic Azad University https://orcid.org/0000-0002-8173-3526
mohsen torabian
Takestan Islamic Azad University

Research Article

Keywords: Bursts, Resource management, Energy efficiency, SLA violation, Overloaded and underloaded hosts

Posted Date: February 14th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-836197/v1

DBSHD: Dynamic Burst-aware Source Host Detection in the Cloud

Somayeh Rahmani [a], Vahid Khajehvand [b,*], Mohsen Torabian [c]

[a]*Department of Computer Engineering, Takestan Branch, Islamic Azad University, Takestan, Iran*

[b]*Faculty of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*

[c]*Department of Mathematics, Takestan Branch, Islamic Azad University, Takestan, Iran*

Corresponding Author Email:vahidkhajehvand@gmail.com

Abstract

One of the essential phases during the Virtual Machine (VM) consolidation is to detect the source (underloaded and overloaded) hosts. Bursts with a sudden and transient increase in the workload have an adverse effect on this process. Also, burst-aware methods for detecting the overloaded and underloaded hosts play a key role in achieving performance and energy efficiency tradeoffs in the cloud computing systems. In this study, we deal with these methods as an open issue in the resource management process. We present a dynamic burst-aware method for detecting the overloaded and underloaded hosts. The principal idea in our proposed method is to keep the weighted average load of the hosts within a dynamic range. The thresholds and weights (impact factors) are obtained from the analysis of the historical data related to hosts. Furthermore, we have used the CloudSim simulator and PlanetLab real-world dataset in order to measure the performance of our proposed method. The simulation results verify the superiority of the proposed technique in achieving performance and energy efficiency tradeoffs, compared to the benchmark methods.

Key Words: Bursts; Resource management; Energy efficiency; SLA violation; Overloaded and underloaded hosts

1. Introduction

With the rapid growth of Internet technology, computing resources have become cheaper, stronger, and more accessible than ever before. These changes in technology have resulted in realizing a novel paradigm for computing, which is referred to as the "cloud computing" [1]. Even though cloud computing provides fast, dynamic, and reliable services for its users, it still faces many challenges [2, 3]. Considering the complexity and scale of cloud computing systems, performance, and energy efficiency are two key challenges that require special investigations.

A cloud is defined as a place residing on the network infrastructure where the computing resources and information services are readily accessible and easily usable. Therefore, cloud customers can access these resources and services without knowing their exact location [4]. However, since the users' requirements change over time, a major problem in cloud resource management is to predict their requirements. Unpredicted user workloads may lead to the overload or underload phenomenon in hosts which in turn, might result in Service Level Agreement (SLA) violation, inefficient energy management, and cost increase [5]. Accordingly, dynamic VM consolidation is a major technique being used for resource management in cloud computing systems [4, 6-9].

Beloglazov et al. divided the problem of VM consolidation into four sub-problems including (i) Identifying overloaded hosts which need some of their VMs to migrate to other hosts to avoid the service failures, (ii)

Identifying underloaded hosts which require all of their VMs to migrate to other hosts to prevent the energy efficiency degradation and the waste of resources, and then switching those hosts to the states with low power, (iii) Selecting several VMs within an overloaded host to prevent violation of the SLA, and (iv) Locating a new destination for the VMs which are selected for migration from the source hosts [10]. Therefore, identifying the source hosts is a critical issue in the VM consolidation process [6]. Khan et al. in [11] have named the identification of underloaded/overloaded hosts as the selection of source hosts. The selection of source hosts means identifying the overloaded and underloaded physical machines in which the migration process should be performed on them [10].

The primary objective of the VM consolidation is to utilize the live migration technique periodically to decrease the number of required hosts and minimize the SLA violation [12]. To do this job, we should identify underloaded and overloaded hosts for such a migration. The migration of the VMs is the most important component used in the VM consolidation process to minimize the count of servers to reduce the energy usage [9, 12] and manage the service failures proactively [8]. In other words, this technique aims to reduce the energy usage of the data centers by turning the underloaded hosts off and prevent the SLA violation in the overloaded hosts. Dynamic VM consolidation can be based on the static or dynamic thresholds. In dynamic VM consolidation based on static threshold, fixed values are used to identify source hosts. Since the user's requirements are dynamic, using static thresholds is not suitable for dynamic cloud environments [13, 14]. Therefore, in this study, we have proposed adaptive thresholds to determine overloaded and underloaded hosts. The thresholds adapt to the changes in the workload.

When we increase the frequency of VM consolidation, the energy consumption in the data centers is decreased. On the contrary, aggressive VM consolidation leads to adverse effects in the performance of the cloud computing systems [15], because it increases the probability of SLA violation. As a result, the energy efficiency and performance change in opposite directions. Achieving a trade-off between the two metrics is essential from several points of view, such as the operational costs, Quality of Service (QoS), and revenue [3]. Identifying overloaded and underloaded hosts plays a significant role in achieving a trade-off energy-performance and it is an open issue in the process of virtual resource management [16].

Bursts with a sudden increase in the workload size have destructive effects on the VM consolidation process. The majority of studies offering methods for identifying the overloaded and underloaded hosts are sensitive to bursts or workload fluctuations. In these methods, when an overloaded VM is detected, VMs immediately migrate to avoid the SLA violations. However, this situation might occur due to workload explosions or bursts, which are transient and only last for a small amount of time. These bursts reduce the performance and the energy efficiency of the system by increasing the migration counts, SLA violation, and Consolidation Ratio (CR) [17]. The CR is the percentage of the hosts which are turned off during the VM consolidation. In a previous effort [17], we proposed a static burst-aware algorithm called Random Early Detection of the Migration Time (REDMT) to identify the source hosts. In this study, we present a dynamic burstiness-aware method to identify source hosts. Our proposed method attempts to keep the servers in the normal state by maintaining their mean load in a dynamic range. We propose a new method for calculating the average load of physical machines. The average and range are calculated by analyzing the historical data related to the hosts. The results of the experiments indicate that, compared to the benchmark methods, the proposed algorithm is efficient from the viewpoints of energy consumption and performance. The main contributions of this study are as follows:

1- We provide a comparison of the different studies for host overload/underload detection.
2- We present a dynamic burst-aware method to identify the overloaded and underloaded hosts.
3- We define adaptive thresholds using the statistical analysis of the historical data related to the hosts to determine the source hosts.
4- We propose a novel weighting method for calculating the weighted average load of the hosts.

This rest of the paper is organized as follows: The overview of the literature is provided in Section 2. Section 3 defines the problem. We provide the proposed algorithm in Section 4. Section 5 describes metrics and the setup used for the simulation. Section 6 interprets the simulation and statistical analysis results. In the end, we summarize the conclusions and suggestions for further research in section 7.

2. Lecture review

In recent years, various studies have investigated energy efficiency in cloud data centers [6, 10, 18-30]. The majority of the existing VM consolidation methods only focus on balancing the trade-off between the SLA performance desired by the customers and the energy consumed by the servers [6, 24, 31]. To the best of our knowledge, there are very limited studies on the VM consolidation which are not sensitive to the workload fluctuation. Bursts or workload fluctuations decrease the performance of the cloud computing systems by increasing the probability of SLA violation, count of migrations, and the CR metrics [32]. In this section, we review the literature concerning the consolidation of VMs in general and the identification of the underloaded and overloaded servers in particular. The terms "server", and "host" are used interchangeably or concurrently in this study.

Farahnakian et al [23] proposed a proactive technique relying on the Linear Regression-Based CPU Usage Prediction (LiRCUP) algorithm to detect the overloaded hosts. This algorithm mainly aims to approximate the short-term future of CPU usage by applying the historical data of the VMs in each host to decrease the violation of SLA and energy consumption. Also, using the k-nearest-neighbor regression algorithm [24], they introduced a method to predict resource utilization and thus to determine the migration time. Based on their findings, the proposed technique resulted in a decrease in SLA violation and energy usage.

Beloglazov and Buyya [33] suggested a reactive method for dynamic VM consolidation using the dynamic thresholds. The thresholds were computed based on the statistical analysis of the historical data obtained from the VMs to guarantee a desirable coverage of the Service Level Agreements (SLA). In another work [10], adjustable and adaptive heuristics were proposed for energy saving and increasing the performance of the dynamic consolidation. They utilized reactive and proactive procedures for the detection of the overloaded host, including Median Absolute Deviation (MAD) and Inter Quartile Range (IQR) in the reactive procedure, as well as Local Regression (LR) and Local Regression Robust (LRR) in the proactive procedure. The usage history of CPU was employed to predict the load of the host in proactive methods. Finally, they proposed using Random Selection (RS), Minimum Utilization (MU), Minimum Migration Time (MMT), and Maximum Correlation (MC) algorithms for selecting the VMs.

Shaw and Singh [34] suggested a method for detecting the overloaded hosts with the purpose of decreasing the count of migrations and making a tradeoff performance-energy. This method uses exponential-smoothing time-series based on historical data to forecast the overloaded hosts. They make decisions for migration and placement of the VM based on the prediction of the CPU utilization. When detecting an overloaded host, they perform no immediate migration, since this condition might be due to a burst. Therefore, the migration takes place if such a condition continues for a while.

A novel approach for decreasing the execution time of the resource management process in the cloud computing systems was developed by Arianyan et al. [18]. The variant heuristics were applied in the mentioned approach for detecting the underloaded hosts and also the VM placement sub-problem based on multi-criteria decision-making. They propose three methods based on the heuristics in order to detect the underloaded hosts. Improving energy efficiency, decreasing the migration count, and minimizing the violation of SLA are some of the objectives in their study. They offer some other techniques in another study to optimize energy efficiency, minimize the SLA violation, and lower the count of migrations via presenting several approaches in order to active management of the resources. In the mentioned study, they try to solve the problem of determining the overloaded hosts and selecting the VMs for the migration. They suggest a window-moving average policy to identify the overloaded hosts using multiple resources (i.e., CPU, RAM, and Bandwidth). Additionally, they propose a multiple-criteria decision-making technique with the purpose of choosing VMs from the overloaded hosts to eliminate hotspots and reduce the SLA violation [19].

In the context of virtual resource scheduling, Zhu et al. [35] suggested a three-dimensional method. They divided the virtual resource management into three phases including: allocation, scheduling, and optimization. They further designed different algorithms for the phases mentioned above. The virtual resource scheduling phase is the same as the VM consolidation process. In this phase, they utilized auto-regressive and time-series prediction methods to propose a strategy for multi-dimensional host overloading detection to prevent the

frequent migration of VMs. This strategy takes the usage of memory, CPU, and network bandwidth into account.

Li et al. [7] presented a method for VM consolidation using the Bayesian network technique. They proposed an Estimation of the Overload Probability (EOP) algorithm by considering the current resource usage and overload probability in the current hosts to identify the overloaded hosts. The EOP algorithm estimates the host's overload probability. If this probability is bigger than an adaptive threshold, this host is labeled as an overloaded host. Luo et al. [36] suggested a dynamic resource management scheme to improve the VM consolidation process using meta-heuristic algorithms. For this purpose, they used the ~~of~~ modified shuffled frog leaping algorithm. They focused on reducing SLA violation and energy consumption.

Li et al. [31] suggested a model for balancing service quality and energy usage in data centers. To do this, they define a utility function and try to minimize it. This function contains two sub-functions, one to determine the level of user satisfaction regarding the QoS and the other to calculate the energy consumption. The value of this utility function is obtained by dividing the two sub-functions. In the model, the Euclidean distance is used to compute the user satisfaction and total energy consumption. They consider throughput, response time, and cost as the quality of service requirements and the disk and CPU utilization as the resources for calculating the energy consumption.

Naeen et al. [37] suggested a stochastic-process based approach for VM consolidation in the case of varying workloads. In their approach, they attempt to minimize the costs by targeting the parameters such as SLA violation, energy usage, the count of migrations, and the switching of the hosts. Also, they suggest methods for identifying the source hosts and VM placement in the VM consolidation process based on the stochastic process.

Melhem et al. [28] presented a method for identifying the source hosts using the Markov chains. In this method, they determine the current state of hosts by comparing the CPU usage with some underload and overload thresholds. Further, they predict the future load of the hosts using the proposed Markov chain. Then, they determine underloaded and overloaded hosts by considering the future and current load of the hosts.

Rahmanian et al. [38] introduced a cost-efficient and penalty-aware method in order to detect the underloaded hosts. In this method, they define a function for calculating the penalties based on the amount of SLA violations of the VMs allocated to hosts. In this method, hosts are considered as underloaded hosts that have the least amount of the function. Yadav et al. [39] presented dynamic energy-aware algorithms by using linear regression for identifying overloaded hosts. These algorithms try to minimize the energy consumed by the hosts and SLA violation of the cloud computing systems.

Sayadnavard et al. [40] proposed a VM consolidation approach which considers the reliability of the hosts in this process. This reliability is modeled using the Markov chains. Based on this model, they propose several algorithms for different phases of VM consolidation. Furthermore, they calculate the probability of failure in hosts in order to determine the migration time, and considering the results, they make decisions for migration. They aim to establish a tradeoff energy-reliability in the cloud computing systems.

Focusing on increasing energy efficiency, Aryania et al. 2018 [41] introduced a meta-heuristic algorithm to solve the VM consolidation problem by applying the ant colony meta-heuristic method. Considering a reduction in the number of active hosts as one of the factors to increase energy efficiency, they aimed to turn off more hosts in their proposed method. They also considered the energy consumption as one of the main factors for migration of VMs in this process. Rahmani et al. [42] suggested a burstiness-aware VM consolidation approach to improve the efficiency in data centers. The approach utilize of the static thresholds for identifying the source hosts.

Fard et al. [25] presented a temperature-aware consolidation method that utilizes dynamic thresholds for identifying the source hosts as a means to obtain a tradeoff between energy consumption and quality of service. Alsadie et al. [43] also proposed a fuzzy logic approach to identify source hosts. Similarly, in another work, Abdelsmea et al. [44] used a multiple regression algorithm that could utilize CPU usage, amount of memory, and bandwidth to detect the overloaded hosts. The algorithm is found to establish a trade-off SLA violation-energy. In other words, the aforementioned study applied a hybrid regression method for optimizing the functioning of the VM consolidation technique by maximizing the accuracy of the procedure of identifying the

overloaded hosts. They used a multiple regression method to identify overloaded hosts. El-Moursy et al. [22] improved the algorithm presented in [44].

Using a fractal model, Yang et al. [45] suggested a predictive algorithm to specify the migration time. They considered the migration of VMs as a way to reach a balanced load. In their study, the load monitoring module has been applied for obtaining the source usage data of the VMs; and by applying a scheduling module, VMs were selected from the overloaded hosts for migration. Moreover, the VMs do not migrate instantly so as to reduce the effects of bursts or sudden explosions. Nonetheless, migrations are performed if the host is marked as overloaded k times. Li et al. [46] presented an algorithm based on robust simple linear regression for source host detection. They also proposed an adaptive threshold to determine underloaded hosts.

Further, Horri et al. [47] present a method to calculate the utilization of the host in order to identify the underloaded hosts. In the presented technique, the CPU usage is defined as a weighted function proportional to the ongoing load of the host and the count of VMs. In this method, the host with the lowest amount of CPU usage is considered as an underloaded host. In case, all the VMs in the underloaded host could migrate, then the host would be switched off [47]. By utilizing the RAM and CPU usage data Castro et al. [21], introduce a new model for calculating hosts energy consumption. Also, they suggest a method for VM placement using the proposed model. Furthermore, a proactive method is proposed to detect underloaded hosts which utilizes an algorithm based on the exponential weighted moving average technique.

Table 1 contains a comparison of the related literature and underlines the characteristics of the current research. As it is shown, there are few studies such as [7, 17, 34, 42], which, detection of underloaded and overloaded hosts is not sensitive to the workload explosions. Detection of these hosts, while considering the bursts, plays a key role in energy saving and performance improvement in the cloud computing systems. Therefore, we investigated the issue mentioned above in this study, since establishing a trade-off energy-performance is one of the main challenges in resource management. The algorithms proposed for determining the underloaded and overloaded hosts are often sensitive to the workload burst. In these algorithms, VMs migrate as soon as the hosts are overloaded. However, when we increase the number of migrations and the consolidation ratio, the energy consumption of the system is increased and its performance is degraded. Therefore, due to the transiency of workload bursts, the algorithms should not be sensitive to them. Our proposed dynamic algorithm attempts not to be sensitive to the bursts while minimizing their negative effects on the performance of the cloud computing systems. Our proposed dynamic burst-aware algorithm, unlike the REDMT algorithm, uses dynamic thresholds and novel weighted average to identify source hosts. The thresholds and weights are determined by the statistical analysis of historical data related to hosts. In addition, the results obtained from the simulations confirm the benefits of utilizing the proposed algorithm over other benchmark methods.

Table 1. comparison of the related works

Publications	**Source host selection**				**Concerns**						**Experiment**			
	ST	**DT**	**DOH**	**DUH**	**BA**	**SLAV**	**NM**	**RW**	**EE**	**CR**	**S**	**RD**	**SD**	**T**
Farahnakian et al. (2013) [23]	*		*			*			*		*	*		
Farahnakian et al. (2013) [24]	*		*			*			*		*	*		
Beloglazov and Buyya (2010) [33]	*		*	*		*	*		*		*	*		
Beloglazov and Buyya (2012) [10]	*	*	*	*		*	*		*		*	*		
Shaw and Singh (2015)[34]	*		*		*	*	*		*		*	*		
Shaw and Singh (2015)[34]	*		*		*	*	*	*		*	*	*		
Luo et al. (2014) [36]	*		*			*			*		*	*		
Fard et al. (2017) [25]		*	*	*		*	*			*	*	*		
Abdelsamea et al. (2017) [44]	*		*			*	*		*		*	*		
Arianyan et al. (2015) [18]	*			*		*	*		*		*	*		
Castro et al. (2016) [21]		*		*		*			*		*	*		
Li et al. (2017) [7]		*	*			*	*		*		*	*		
Zhu et al. (2017) [35]	*		*	*		*			*		*	*		
Yang et al. (2012) [45]	*		*	*	*		*	*	*					*
Melhem et al. (2018) [28]		*	*	*		*	*		*		*	*		
Li et al. (2017) [31]	*		*	*		*	*		*		*		*	

Rahmanian et al. (2017) [38]	*			*		*	*		*		*	*			
Sayadnavard et al. (2019) [40]	*		*	*		*	*	*	*		*	*			
Horri et al. (2014) [47]		*		*		*	*		*		*	*			
Aryania et al. (2018) [41]		*	*	*		*	*			*	*	*			
Yadav et al. (2018) [39]		*	*			*	*		*		*	*			
El-Moursy et al. (2019) [22]	*		*			*	*		*						
Li et al. (2019) [46]		*	*	*		*	*		*	*	*	*	*		
Alsadie et al. (2019) [43]		*	*	*		*	*		*		*	*			
Naeen et al. (2020) [37]	*		*	*		*	*		*	*	*	*			
Rahmani and Khajehvand (2020) [17]	*		*	*	*	*	*	*	*		*	*			
Proposed work		*	*	*	*	*	*	*	*	*	*	*			

ST: Static Threshold, **DT:** Dynamic Threshold, **DOH:** Determine the Overloaded Hosts, **DUH:** Determine the Underloaded Hosts, **BA:** Burst-Aware, **SLAV:** SLA Violation, **EE:** Energy Efficiency, **NM:** Number of Migrations, **RW:** Resource Wastage, **CR:** Consolidation Ratio, **S:** Simulation, **T:** Testbed, **RD:** Real Data trace, **SD:** Synthetic Data trace

3. Statement of the Problem

The resource management system is an IaaS environment including a large-scale data center with N heterogeneous server indicated as the $H = \{H_1, H_2, \ldots, H_N\}$ set. Each server is described using the amount of CPU utilization , and amount of RAM, disk and network bandwidth. In order to store the data of the different VMs, the data center includes a Network Attached Stored (NAS), while the hosts only employ local disks for loading the operating system. As such, when migrating performing a migration, it is only necessary to transfer the main memory of the VMs. In order to formalize the delivered, the clients sign SLAs with the service provider. Because the resource management system is an IaaS environment, the QoS requirements are considered independent of the workload. In this research, we assume that the QoS requirements are workload-independent. In other words, the intended system is not dependent of the application. In this study, we do not calculate the penalty of the SLA violation, but we do calculate the amount of SLA violation which somehow reflects this value.

Similar to [10], the software layer of the system includes a local manager for each host and a global manager. The local managers in each host monitor the load of the corresponding host and identify the overloaded or underloaded hosts. The local managers utilize the algorithm proposed in the next section to do this job. Normal, overloaded, and underloaded servers are represented using H_{Norm}, H_{Over}, and H_{Under} sets, respectively. The placement of the VMs is made by the global manager. The request belonging to each user is allocated to a VM and is displayed using the $VM = \{VM_1, VM_2, \ldots, VM_m\}$ set. The historical data related to servers is demonstrated using vector based on Eq. (1).

$$\vec{U}_{H_i}^{l_w} = (U_{H_i,1}, U_{H_i,2}, \ldots, U_{H_i,l}) \qquad i \in H \tag{1}$$

To display the placement of VMs to servers, we have used the $A = [a_{ij}]_{n \times m}$ matrix. According to Eq. (2), if the j^{th} virtual machine is assigned to the i^{th} host, this entry equals to 1 and otherwise to 0.

$$a_{ij} = \begin{cases} 1 & VM_j \quad is \quad in \quad H_i \\ 0 & otherwise \end{cases} \tag{2}$$

The goal is to provide services for different users with the least energy consumption, violations of the SLA and consolidation ratios. For this purpose we have used the ESC metric suggested in [17]. Consequently, the objective function is defined as follows

$$\min ESC$$

Subject to:

$$\sum_{i=1}^{n} a_{ij} \leq 1 \qquad \forall j \in VM \tag{3}$$

$$\sum_{j=1}^{m} a_{ij} W(t)_j \le C_i \qquad \forall i \in H \tag{4}$$

$$\min_{th} \le avgload_{t_i} \le \max_{th} \qquad i \in H \tag{5}$$

Constraint (3) specifies that each VM is assigned to only one server at each time. Constraint (4) states that the aggregate amount of resources required for the VM assigned to the i^{th} host must be smaller than that host's capacity. Constraint (5) also specifies that the weighted average load of hosts must be between min_{th} and max_{th}. The min_{th} and max_{th} are the dynamic lower and upper thresholds, respectively. How to calculate the weighted load and the thresholds are described in the following section. A summary of the symbols being used in this study is provided in Table 2.

Table 2. The symbols used in our study for modeling

Symbol	Definition
S	Set of the servers
VM	Set of the virtual machines
H_{Normal}	Set of the hosts in the normal state
H_{Over}	Set of the hosts in the overload state
H_{Under}	Set of the hosts in the underload state
i	Index of hosts
j	Index of virtual machines
p	Index of historical data
$\vec{U}^{l}_{H_i}$	Historical data of the $host_i$
A	The matrix of assigning VMs to hosts
$W_j(t)$	The amount of resources requirement of VM_j at time t
C_i	The capacity of $host_i$
max_{th}	Adaptive overload threshold
min_{th}	Adaptive underload threshold
Avg_load_{lw}	The average load of a host during the historical data vector at time t
$Current\_load_t$	the current load of the host at time t
Avg_load_t	The average load of a host at time t
l_w	Length of historical data
W_{lw}	Impact factor of the historical data
W_{new}	Impact factor of the host's current load
Avg_total	Average of the historical data and current load of the host
Avg_{lw}	average of the historical data
D_{lw}	Distance of the historical data to the total average
D_{new}	Distance of the current load of the host to the total average
D_{total}	Sum of distances
ADM_i	Average Deviation of Mean metric for $host_i$
Ta_i	Total time that $host_i$ has been active
Cr_j	The total CPU capacity requested by the VM_j during its lifetime
Cd_j	Performance degradation of the VMj because of migration
NTO	The number of hosts that turning off at any time interval of the VM consolidation process

4. Proposed Method

In this section, we will describe our proposed method. First, we describe the proposed algorithm for determining underloaded and overloaded hosts, and then we describe the proposed method for calculating the adaptive lower and upper thresholds.

4.1. Proposed Algorithm

Detection of overloaded and underloaded hosts is a major sub-problem in the VM consolidation process. To solve this, we present a proactive method in the current study. In our proactive method, before a host becomes

overloaded or underloaded, some of its running VMs are migrated to other hosts which are in the normal state. This job is done to prevent SLA violation and energy wastage. The algorithm periodically monitors loads of the hosts and in the case of detecting the underloaded and overloaded hosts, the VM monitor migrates the appropriate VMs.

As previously stated, the present study aims to propose a dynamic burst-aware algorithm for determining the source hosts. To do this, the idea of the REDMT algorithm is employed to manage the load of the hosts in the cloud computing systems, which helps to establish a trade-off performance-energy. The REDMT algorithm is a static burst-aware algorithm to determine the source hosts. The pseudo-code of the (Dynamic Burst-aware Source Host Detection) DBSHD algorithm is illustrated in Algorithm 1. Since the burst reduces the efficiency of cloud computing systems, the DBSHD algorithm uses the weighted average load of hosts to detect the overloaded and underloaded ones. It leads to the insensitivity of the algorithm to the sudden explosions or transient bursts. The DBSHD keeps the weighted average load of the hosts between two lower and upper thresholds. Static thresholds are unsuitable for a dynamic and real cloud environment in which the workload is unpredictable and changes with the time. As a result, we have utilized adaptive thresholds in the DBSHD algorithm, which change at any time based on the host's load. We suggest new methods for the automatic adjustment of the upper and lower thresholds using the statistical analysis of workloads collected from ~~of~~ the VMs and hosts. In the following paragraphs, we will explain these thresholds in detail.

The weighted average load of a host is denoted by avg_load_t which is calculated using Eq. (6). The avg_load_{lw} is the average load of historical data of the host. The length of the historical data is indicated by L_w. L_w impacts the efficiency of the algorithm. The appropriate value for the L_w is obtained using the simulation that results are provided in section 6.1. And, the $current\_load_t$ expresses the host's current load at the given moment t. Since whatever distance of the observations to the total average is ~~be~~ lower it's weight should be greater, so we used ~~of~~ the distance to define the weight. In the Eq. (6) w_{Lw} is the weight of avg_load_{Lw} that is defined according to Eq. (9), and w_{new} is the weight of $current\_load_t$ that is defined according to Eq. (10). According to the Eq. (8) avg_total is the average of the historical data and $current\_load_t$. In the Eq. (9) D_{lw} is the distance of the historical data to the avg_total, and according to the Eq. (12) D_{new} is the distance of the $current\_load_t$ of the host to the avg_total.

$$avg\_load_t = w_{l_w} \times avg\_load_{l_w} + w_{new} \times current\_load_t \tag{6}$$

$$avg\_load_{l_w} = \frac{\sum_{p=1}^{l_w} U_{H_i,p}}{l_w} \qquad i \in H \tag{7}$$

$$avg\_total = \frac{\sum_{p=1}^{l_w} U_{H_i,p} + current\_load_t}{l_w + 1} \qquad i \in H \tag{8}$$

$$w_{l_w} = 1 - \frac{D_{l_w}}{D_{new}} \tag{9}$$

$$w_{new} = 1 - \frac{D_{new}}{D_{total}} \tag{10}$$

$$D_{l_w} = |avg\_load_{l_w} - avg\_total| \tag{11}$$

$$D_{new} = |current\_load_t - avg\_total| \tag{12}$$

$$D_{total} = D_{new} + D_{lw} \tag{13}$$

The thresholds are adaptively determined based on the statistical analysis of the historical data or state of the hosts. We will explain the method of calculating the thresholds in the next sections. If the avg_load_t a host is not within the specified range, then migration occurs. In this study, the load has the same meaning as CPU utilization.

Algorithm 1: Dynamic Burst-aware Source Host Detection (DBSHD)

1. calaulate avg_load_t for host
2. calculate ADM(utilization_history) for host
3. max_{th}= 1- S_{upper}*ADM
4. if $avg\_load_t > max_{th}$ then
5. host state is overload and some VMs
6. must migrate
7. end_if
8. calculate min_{th} due to Eq. (16) or Eq. (17)
9. if $avg\_load_t < min_{th}$ then
10. host state is underload and all VMs must
11. migrate, and turn it off
12. else
13. host state is normal and migration is not needed
14. end_if

We consider three states for hosts: normal, underload, and overload. First, avg_load_t is computed for all the active hosts (line 1). Then the upper threshold is calculated using the Eq. (14). If the avg_load_t of the host is higher than the max_{th}, we conclude that the host is overloaded (lines 3-7). Next, the lower threshold (line 8) is estimated using Eq. (15) or Eq. (16). If the avg_load_t of a host is lower than the min_{th}, then we conclude that the host is underloaded. However, if it is between two thresholds, the host is in the normal state (lines 9-14). Figure 1 describes the operation of the DBSHD algorithm. This figure is composed of two main parts: calculating the thresholds and $load\_avg_t$. By calculating these two metrics, we determine the state of hosts. Since the load of VMs allocated to a host changes with the time, the state of hosts could vary among the three mentioned states. The methods used for calculating thresholds will be described in the next sections.

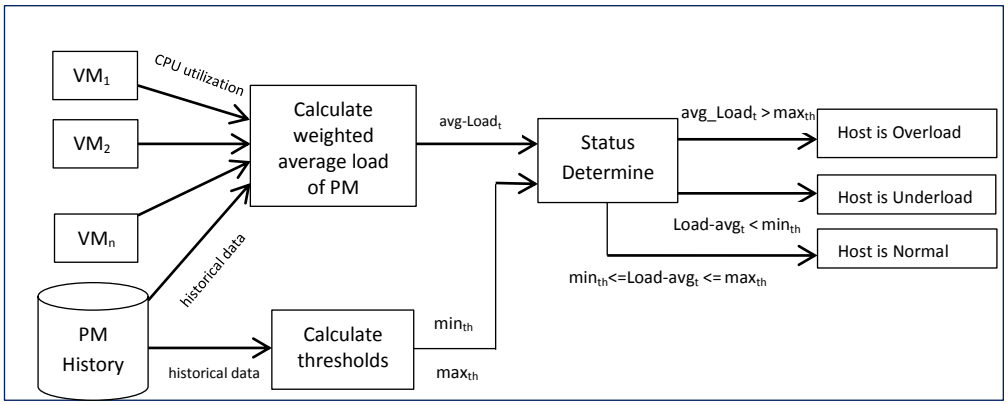

Figure. 1 DBSHD algorithm

When workload fluctuations occur, the hosts become underloaded or overloaded and may cause the waste of the energy or SLA violation. Therefore, the DBSHD algorithm is implemented to keep the hosts in the normal state. If the avg_load_t of the host is greater than max_{th}, it means the host is in the overloaded state. The algorithm reduces the hosts' load by migrating several of its VMs to prevent the SLA violation. When the avg_load_t of the host is lower than min_{th}, it means the host is in the underloaded state. In order to reduce the waste of resources and save the energy, all the VMs allocated to it are migrated and the corresponding host is turned off. Otherwise, the host is assumed to be in the normal state and no migration will occur in this case.

It is worth mentioning that insensitivity to bursts or workload explosions of the hosts is one of the main characteristics of the DBSHD algorithm. The algorithm relies on this characteristic to avoid unnecessary migrations as well as repeatedly switching the hosts to the sleep mode because of the bursts. (One of the most important features of the DBSHD algorithm is insensitive to bursts or the workload explosions of hosts. Employing this feature, the algorithm avoids frequent switching of the hosts to the sleep modes and unnecessary migrations caused by the bursts). Also, by maintaining the avg_load_t of the host between the two adaptive thresholds, while increasing the energy efficiency and performance, the overhead of the cloud computing systems is reduced. In fact, by considering the avg_load_t of a host, instead of focusing on momentary loads, the algorithm focuses on the trend of the load. (In other words, considering the avg_load_t of a host, we have focused

on the load trend instead of the momentary load.) In fact, using the adaptive thresholds and impact factors to determine the source hosts is performed more accurately and thus, fewer SLA violations occur.

4.2. Calculating Adaptive Upper Threshold

We have used the Average Deviation of Mean (ADM) metric for definition the upper threshold. For i^{th} host, the ADM metric is defined according to Eq. (14). As mentioned before, the $current\_load_t$ expresses the host's current load at the given moment t, avg_total is the average of the historical data and $current\_load_t$, and $U_{i,p}$ represent the p^{th} member of the vector historical data of the i^{th} host. Finlay, the upper threshold is defined according to Eq. (15).

$$ADM_i = \frac{\sum_{p=1}^{l_w} |U_{H_i,p} - avg\_total_i| + |current\_load_t - avg\_total_i|}{l_w + 1} \qquad i \in H \tag{14}$$

$$\max_{th} = 1 - S_{upper} \times ADM \tag{15}$$

In Eq. (15), $S_{upper} \in R^+$ is a parameter which determines the consolidation intensity in the system. The parameter S_{upper} allows the safety adjustment of the method and larger values for S_{upper} cause less SLA violation, but increase the energy consumption of the cloud computing systems.

4.3. Calculating Adaptive Lower Thresholds

We propose two methods for calculating the lower threshold using analysis of the historical data and the state of the hosts in the cloud computing systems, and we will discuss their details in the following sections.

4.3.1 Calculating Adaptive Lower Threshold Using Mean

The main idea behind this method is to consider the mean load of all the hosts (except for the overloaded hosts) in the cloud computing systems for calculating the lower threshold value. The method for calculating this mean is displayed in Algorithm 2. At first, the overloaded hosts must be determined, and to do this, a remain_hosts set is created which includes no overloaded hosts (line 1). Then, the mean load of all hosts in the set (mean_r) is calculated (lines 3-6). Finally, the lower threshold is computed (line 7).

Algorithm 2 : Calculate the mean load of hosts with the exception of overloaded hosts

```
1.  remain_hosts = H - H_over
2.  // calculate mean for remain_hosts set
3.  for all hosts exist in remain_hosts
4.      sum+= sum + current_load_t;
5.  end_for
6.  mean_r= sum/ number of remain_hosts
7.  min_th = 1 - S_lower* mean_r
```

When mean_r is low, it indicates that the hosts' load is relatively low, and we can turn off more hosts. Therefore, we need a threshold with a larger value. However, when mean_r is high, it indicates that the hosts' load is relatively high. Therefore, in order to keep more hosts in the normal state, we should switch fewer hosts into low-power modes. As a result, we need a threshold with a lower value. The lower threshold is defined based on Eq. (16).

$$\min_{th} = 1 - S_{lower1} \times mean\_r \tag{16}$$

In Eq. (16), $S_{lower1} \in R^+$ is a parameter that defines the safety of the method, similar to the parameter S_{upper} in the section 4. 2. Also, mean_r is the average load of all hosts existing in the set remain_hosts.

4.3.2. Calculating Adaptive Lower Threshold Using the First Quartile

In this section, we describe the second proposed method for calculating a dynamic lower threshold. We use the first quartile of the historical data of a host to calculate the lower threshold. This quartile is a value in the

historical data which 25% of the values lie below it. The formula for computing the lower threshold is represented in Eq. (17).

$$\min_{th} = (1 - Q1) \times S_{lower2} \tag{17}$$

In Eq. (17), Q1 is the first quartile of the host's historical data. Furthermore, S_{lower2} parameter, similar to S_{upper} and S_{lower1} parameters, measures the consolidation intensity. However, an increase in the value of this parameter increases SLA violation and reduces energy consumption.

5. Experiment Setup and Metrics

We use the CloudSim simulator to measure the efficiency of the proposed technique. The CloudSim simulator is an extendable and programmable software, developed by Calheiros et al. in 2011[48]. and is open-source. This adaptable simulator enables developers to easily model the large-scale virtualized systems, such as the CSS. In our experiments, we have designed a cloud establishment with 800 PMs. We have chosen two configurations for them. Half of these hosts are HP ProLiant ML110 G4(Intel Xeon 3040, 2 cores _ 1860 MHz, 4 GB. The other half includes HP ProLiant ML110 G5(Intel Xeon 3075, (2 cores _ 2660 MHz, 4 GB) systems. The features of PMs are displayed in Table 3 [15].

Table 3. PMs' Characteristics [10]

Server	CPU model	Cores	Frequency(MHz)	RAM(GB)
HP ProLiant G4	Intel Xeon 3040	2	1860	4
HP ProLiant G5	Intel Xeon 3075	2	2660	4

Moreover, the VM types in the experiments are defined based on the specifications of the Amazon EC2 instant types, except that in the current study, all the VMs are single-core VMs. This is because the workload data to be used in the simulation will be obtained from single-core VMs. Table 4 presents the specifications of the VM types. The experiments utilize four different VM types as follows: Extra-Large Instance (2000 MIPS, 3.75 GB); High-CPU Medium Instance (2500 MIPS, 0.85 GB); Small Instance (1000 MIPS, 1.7 GB); and Micro Instance (500 MIPS, 0.613 GB).

Table 4. VM types' properties (four Amazon EC2 VM types)

VM size	CPU(Mips)	RAM(GB)
Micro Instance	500	0.613
Small Instance	1000	1.7
Extra Large Instance	2000	3.75
High-CPU Medium Instance	2500	0.85

5.1. Workload Data

For the results to be more realistic, we have to utilize real workload traces in our simulations. As a result, we have employed the data which is provided by the CoMon project. The CoMon project is a monitoring infrastructure designed for PlanetLab [49]. In our experiments, the CPU usage data of nearly 1000 virtual machines has been used; where these VMs had been run in almost 500 different locations around the world. In our experiment, we measure CPU utilization of VMs every five seconds and maintain them in separate files. Then, from the PlanetLab workload traces gathered within March and April 2011, we randomly select ten days. The features of these data are displayed for each individual day in Table 5.

Table 5. Characteristics of the data[10]

Date	Number of VMs	Mean	St. dev.	Median
2011/03/03	1052	12.31%	17.09%	6%
2011/03/06	898	11.44%	16.83%	5%
2011/03/09	1061	10.70%	15.57%	4%
2011/03/22	1516	9.26%	12.78%	5%
2011/03/25	1078	10.56%	14.14%	6%
2011/04/03	1463	12.39%	16.55%	6%
2011/04/09	1358	11.12%	15.09%	6%
2011/04/11	1233	11.56%	15.07%	6%
2011/04/12	1054	11.54%	15.15%	6%
2011/04/20	1033	10.43%	15.21%	4%

5.2. Power and Energy Model

In [21] a model was offered for the calculation of energy consumption with CPU and RAM utilization. Some studies demonstrate that there is a linear relationship between the power consumption of physical machines and the amount of CPU usage. We utilize real data trace about power consumption provided by the results of the SPECpower benchmark [50]. In Table 6, the characteristic of the servers in various modes is shown.

Table 6. Power consumption of Hosts[10]

Host	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP ProLiant G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP ProLiant G5	93.7	97	101	105	110	116	121	125	129	133	135

Changes in the workload result in the changes in CPU usage of the hosts as time passes. Therefore, utilization of the CPU is a time-dependent parameter and is demonstrated using u(t). As shown in Eq. (18), we can define the total amount of energy consumed by hosts by calculating the integral of power consumption over time. The mentioned equation is quite popular in the related literature [10, 18, 19]:

$$E(t) = \int_t P(u(t))dt \tag{18}$$

5.3. Number of Migrations

Virtual machine migration reduces efficiency in cloud computing systems by increasing response time and throughput. In other words, with increasing the number of migrations, efficiency decreases. So reducing the number of migrations is one of the important criteria in determining the efficiency of cloud computing systems.

5.4. Metrics for measuring the SLA violation

Meeting Quality of Service (QoS) requirements for cloud computing systems is very important. QoS requirements are usually provided in the form of SLA and can be determined considering characteristics such as delay, response time and throughput. Since these characteristics depend on the application, it is essential to define a workload-independent metric so that it can be used in the IaaS environment. To do this, we use metrics defined by Beloglazov et al. [10], which are shown down here:

- The percentage of time during which active hosts have reached a 100% CPU usage, that is SLA violation Time per Active Host (SLATAH):

$$SLATAH = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{s_i}}{T_{a_i}} \tag{19}$$

- The overall Performance Degradation due to Migrations (PDM):

$$PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{c_{d_j}}{c_{r_j}} \tag{20}$$

In equation (19), N is the number of hosts; T_{si} is the total time when the S_i has experienced the utilization of 100%, which would lead to a violation in the SLA; T_{ai} refers to the total of the S_i when it is active. In equation (20), M refers to the count of the VMs; C_{dj} shows the estimated value of the degradation of the performance of the VM_j which is caused by migrations; C_{rj} shows the total CPU utilization requested by the VM_j. In our simulations, same as the [10], we set C_{dj} to 10% of the CPU utilization during all the migrations of the VM_j. Since a violation of the SLA is a function of the PDM and SLATAH criteria, same as [10] we use equation (21) to define this relationship:

$$SLAV = SLATAH \times PDM \tag{21}$$

5.5 Performance Metrics

To evaluate the performance of cloud computing systems, we used the ESC criterion according to equation(22), which we defined in our previous work [17]. As can be seen, the effective criteria in ESC include energy consumption, SLAV and CR. More details about the ESC criterion are available in [17]. Furthermore, we have also utilized the ESV metric proposed by Belaglazov et al [10].

$$ESC = E \times SLAV \times CR \tag{22}$$

6. Simulation experiment and result analysis

In this section, we discuss the simulation results. Two DBSHD1 and DBSHD2 methods are introduced to evaluate the DBSHD algorithm in the process of VM consolidation. Also, Eqs. (16, 17) are employed to calculate the lower threshold in the DBSHD1 and DBSHD2 algorithms, respectively. As previously mentioned, the dynamic VM consolidation has four main phases [10]. In this process, we use the DBSHD method for phases 1 and 2, and MMT, and Power-Aware Best Fit Decreasing (PABFD) algorithms proposed in [10] for phases 3 and 4, respectively. In the DBSHD algorithm, we utilize the avg_load_t and ADM variables. The length of the window affects the values of these variables. Therefore, determining the appropriate length for the window is of great importance. In the next section, we determine the appropriate value for this window.

6.1 Length of the Historical Data

We conducted various experiments to determine the length of the window, represented by L_w in this study. We executed the experiments separately on all dates of the PlanetLab workload for the DBSHD1 method with different values of L_w. Also, we incremented the value of L_w from 10 to 40, increasing by 10 each time. The results are shown in Table 7. Due to the objective of our study, which is to establish a trade-off performance-energy, we considered the ESC parameter as a key metric for determining the value of L_w parameter. As shown in Table 7, the ESC metric has the lowest value for L_w = 30. Therefore, the size of the L_w is set to 30 for all the experiments. Figure 2 illustrates the impact of different L_w sizes on the ESC metric for all dates of the PlanetLab workload. In most cases, when the parameter L is equal to 30, ESC has the lowest value.

Table 7. Simulation results for different L_w's (median values for PlanetLab workload)

L_w	Energy(Kwh)	Migrations	SLAV × 10^{-5}	SLATAH%	PDM%	CR ×10^{-2}	ESV × 10^{-3}	ESC × 10^{-5}
10	150.83	4721	0.14	1.26	0.01	0.6	0.20	0.115
20	150	4230	0.14	1.21	0.01	0.56	0.19	0.112
30	**143.46**	**4239**	**0.13**	**1.21**	**0.01**	**0.56**	**0.18**	**0.11**
40	154.35	4288	0.14	1.26	0.01	0.57	0.21	0.123

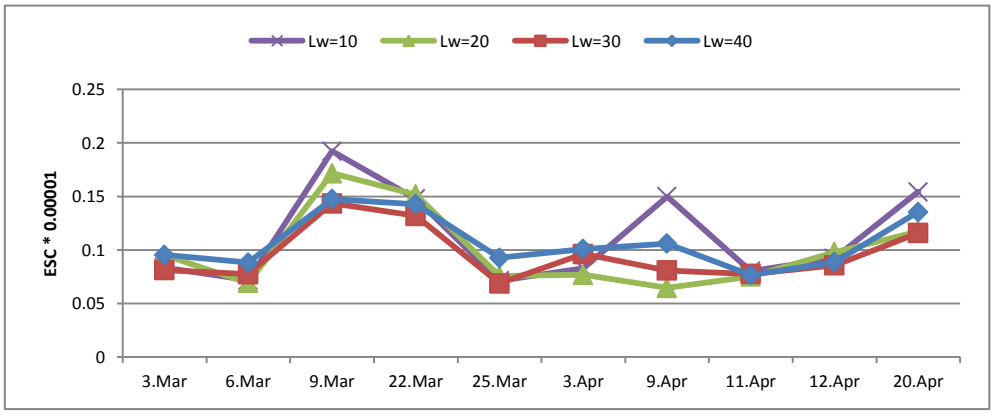

Figure 2. The value of the ESC parameter for different lengths of windows

Similarly, we obtain the safety parameters by simulation. Values of the S_{upper}, S_{lower1}, S_{lower2} are 0.5, 1.5, 0.5 respectively. For each parameter, we changed the values in the following manner: in the case of k, S_{upper} and S_{lower2} parameters, we changed the values from 0.1 to 1.0, increasing by 0.1 each time, and for S_{lower1} we changed the value from 0.5 to 2.5, increasing by 0.5 each time.

6.2. Simulation Results

We compared our proposed methods with other methods, including a combination of some algorithms reported in [10] for VM consolidation phases, i.e. LR, THR, MAD, IQR used for detecting the overloaded hosts. Moreover, we employed the simple method (SM) and the MMT method for detecting the underloaded hosts and VM selection, respectively. We used the PABFD algorithm for the VM placement phase. In [19] and [34], they presented Double Exponential Series (DES) and Windows Moving Average (WMA) algorithms for determining the overloaded hosts, respectively. Similar to the DBSHD algorithm, these algorithms are not sensitive to bursts or workload fluctuations. We also used the REDMT algorithm proposed in [17] for comparison. The LR method was considered as a reference method. As represented in Table 5, the experiments are separately executed for all dates of PlanetLab workload and their median results for the energy use, the migrations counts, SLATAH, PDM, the violations of SLA, ESV, CR, and ESC metrics are provided in Table 8.

Table 8. Simulation results of variant methods for identification of source hosts (median values)

Policy	Energy(Kwh)	Migrations	SLATAH%	PDM%	SLAV× 10^{-5}	ESV× 10^{-2}	CR %	ESC× 10^{-5}
LR	158.27	27418	6.13	0.08	4.62	8.11	2.17	17.00
THR	184.84	25851	5.02	0.07	3.245	6.33	2.47	15.38
MAD	174.50	28969	5.61	0.85	4.735	8.09	2.47	19.90
IQR	185.13	25969	4.94	0.06	2.955	5.75	2.48	14.73
DES	133.62	10899	6.2	0.03	1.95	2.70	1.48	3.87
WMA	153.79	12913	2.76	0.03	0.86	1.36	1.05	1.49
REDMT	135.34	8936	3.75	0.02	0.56	0.85	0.94	0.86
DBSHD1	**143.46**	**4239**	**1.21**	**0.01**	**0.13**	**0.18**	**0.56**	**0.11**
DBSHD2	**136.84**	**5184**	**1.35**	**0.01**	**0.14**	**0.21**	**0.69**	**0.14**

Energy consumption of various methods is depicted in Figure 3. As is can be seen, the DES method has the lowest energy consumption, while the IQR method has the highest energy consumption compared to the other methods. Also, the DBSHD2 and REDMT methods approximately consume the same amount of energy.

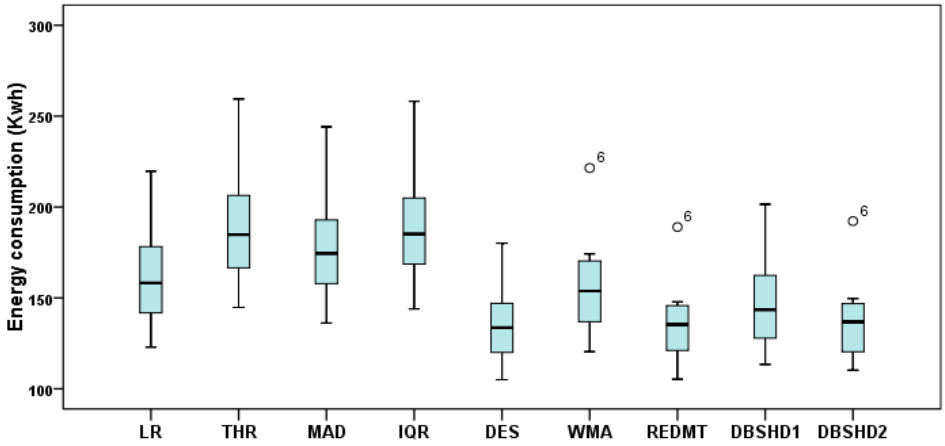

Figure. 3 energy consumption of different methods to determine the source hosts

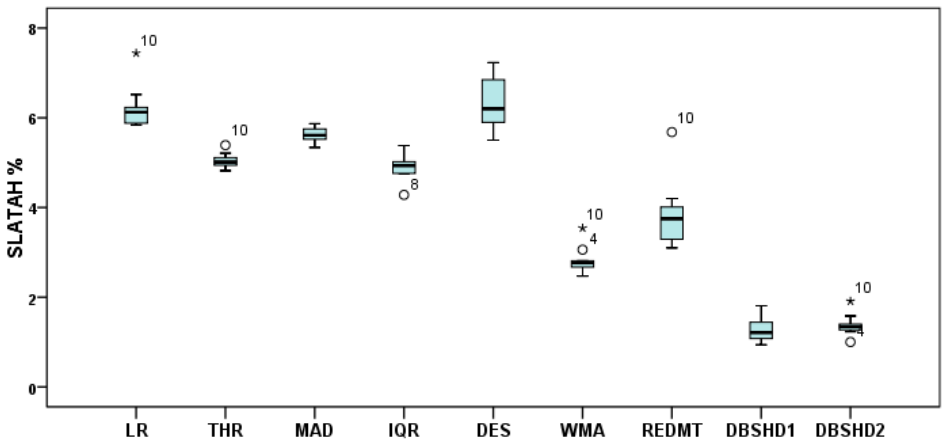

Figure. 4 SLATAH metric of different methods to determine the source hosts

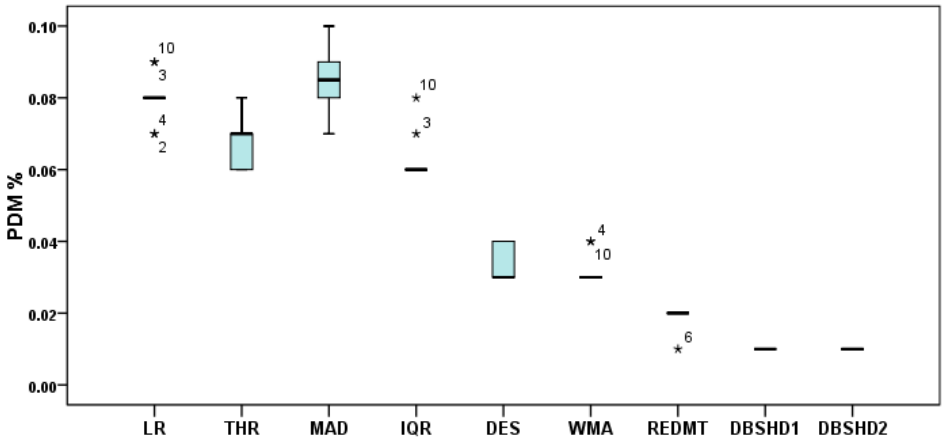

Figure. 5 PDM metric of different methods to determine the source hosts

The load of the hosts in the data centers directly affects the value of the SLATAH metric. This criterion indicates the number of times which the hosts are overloaded. The SLATAH metric of different methods to determine the source hosts is displayed in Figure 4. Our proposed methods outperform other algorithms in terms of the SLATAH metric. Based on the data in Table 8, DBSHD1 and DBSHD2 methods reduce this criterion by 80.26% and 77.98%, respectively, compared to the reference method. Also, the proposed methods compared to the REDMT method reduced the SLATAH criterion by 67.73% and 64%, respectively. This is due to the use of appropriate dynamic upper thresholds to identify the overloaded hosts.

Additionally, the DBSHD1 leads to 80.04% and 56.16% reductions in the SLATAH metric compared to DES and WMA methods, respectively. Figure 5 illustrates the PDM metric of the variant methods to determine the source hosts. Since the proposed methods are insensitive to sudden fluctuations or workload bursts, the number of migrations in our method reduces significantly, compared to the benchmark methods. This leads to a significant reduction in the value of the PDM metric. In addition, the DBSHD1 and DBSHD2 methods lead to an 87.5% decrease in the value of PDM, compared to the reference method, and a 66.66% reduction in the value of PDM metric, compared to DES and WMA methods.

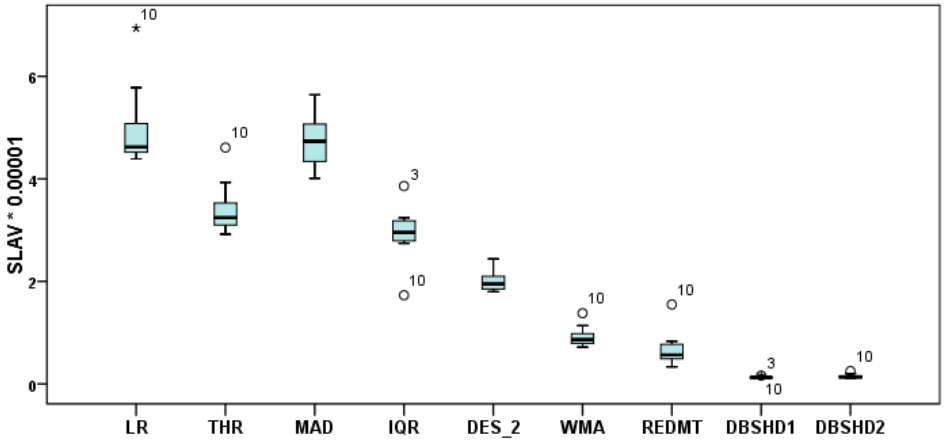

Figure. 6 SLAV metric of different methods to identify the source hosts

The SLA violation is one of the essential parameters for determining the efficiency and satisfaction of the users in cloud computing systems. The amount of SLA violations that occurred in different methods during the VM consolidation process is shown in Figure 6. Due to a significant decrease in the SLATAH and PDM metrics, DBSHD1 and DBSHD2 methods have the lowest amounts of SLA violations, compared to benchmark methods. Also, the value of SLAV in DBSHD1 is less than that of the DBSHD2. The use of dynamic thresholds in the proposed methods has significantly reduced the count of migrations compared to the REDMT method that uses static thresholds. This resulted in a 50% reduction in the PDM metric in the proposed methods compared to the REDMT method. By significantly reducing the count of migrations and preventing the overloaded hosts, the DBSHD1 method reduces the SLA violation by 97.19%, compared to the reference method. Furthermore,

DBSHD1 causes 93.33%, 84.88%, and 76.79% reductions in SLAV metric, compared to the DES, WMA, and REDMT methods, respectively.

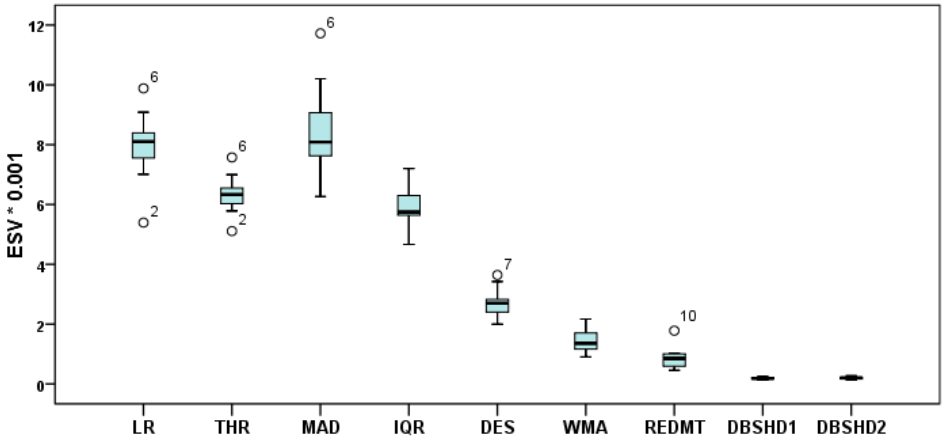

Figure. 7 ESV metric of different methods to identify the source hosts

Figure 7 represents the value of the ESV metric of the different methods to determine the source hosts. As it is shown, DBSHD1 has the lowest amount of the ESV metric. Our proposed methods even outperform the DES on this metric, which has the lowest energy consumption among the benchmark methods. This is due to a significant reduction in the SLA violation (Figure 6). As shown in Figure 7, the DBSHD1 method has the lowest value, compared to the benchmark methods, and the method reduces the ESV metric by 18.23%, compared to the DBSHD2 method. Furthermore, the DBSHD1 leads to 97.66%, 92.96%, 86.03%, and 78.82% reductions in the ESV metric, compared to the reference, DES, WMA, and REDMT methods, respectively.

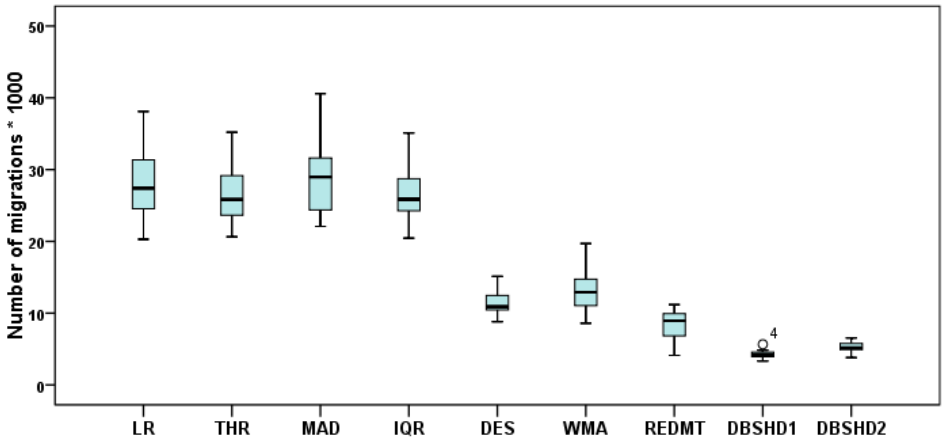

Figure. 8 number of migrations of different methods to identify the source hosts

Figure 8 illustrates the count of migrations performed by different methods. The proposed methods feature the lowest of this parameter. As explained above, the DBSHD1 and DBSHD2 methods prevent repeated migrations using the dynamic thresholds and the average load of the hosts for determining the migration time. As a consequence, these two methods have the lowest count of migrations compared to the benchmark methods. The DBSHD1 method has the lowest number of migrations and leads to 84.54% reduction in the count of migrations compared to the reference method. This method reduces the count of migrations by 61.11%, 67.73%, and 52.56% respectively, compared to the DES, WMA, and REDMT methods. The values of CR metric for various methods are depicted in Figure 9.

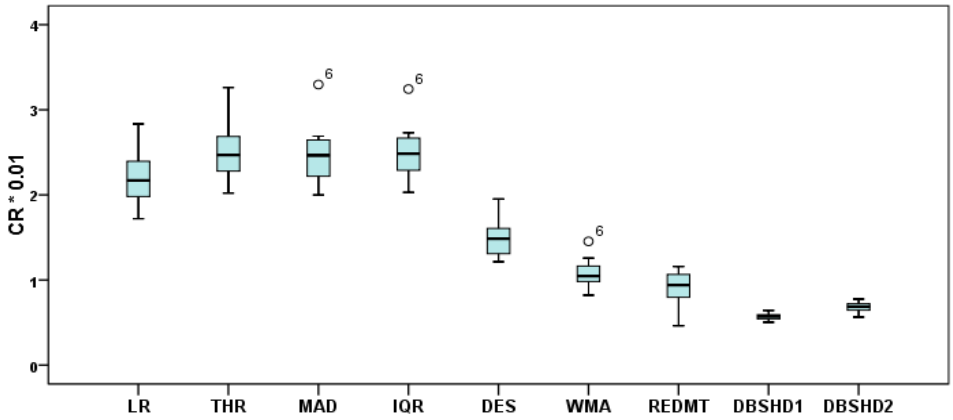

Figure .9 CR metric of different methods to identify the source hosts

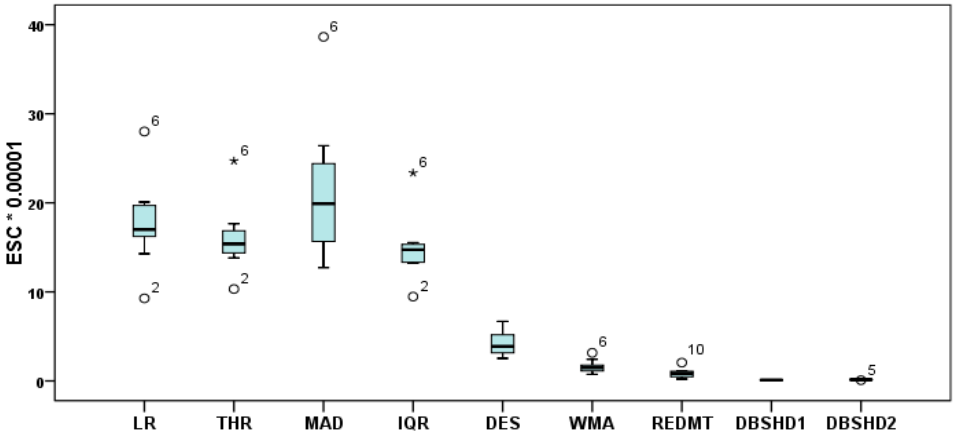

Figure. 10 the ESC metric of different methods to identify the source hosts

We propose the ESC metric in this study since it attempts to establish a trade-off performance-energy. Therefore, a lower value is more suitable for this purpose. A comparison of the ESC metric of different methods to determine the source hosts is displayed in Figure 10. As the figure shows, DBSHD1 and DBSHD2 methods have the lowest values. The ESC metric in DBSHD1 improves by 99.35% by reducing the CR and SLA violations, compared to the reference method (Table 8). Moreover, the DBSHD1 method results in 97.16%, 92.62%, and 87.21% reductions in the ESC metric, compared to the DES, WMA, and REDMT methods, respectively. This is due to the fact that, by maintaining the average load of hosts in a dynamic range, the DBSHD algorithm is indifferent to the fluctuations of the workloads in the VMs. Therefore, the energy efficiency and performance of the system improves simultaneously by a significant reduction in values of CR and SLA violation.

6.3. Statistical Analysis

Statistical analysis is performed in this section to identify the best possible working method. Since our objective is to establish a trade-off performance- energy, we use the ESC metric for the statistical analysis. First, we used the K-S test to determine whether the analysis was parametric or non-parametric. The results of this test are shown in Table 9. According to the data of Table 9, ESC values of all algorithms follow a normal distribution with a *P*-value > 0.05. As a result, in the next step for paired comparisons, we use the T-test.

Table 9. Result of K-S test

Policy	LR	MAD	THR	IQR	DES	WMA	REDMT	DBSHD1	DBSHD2
Significant	0.2	0.2	0.198	0.06	0.2	0.2	0.184	0.2	0.2

We ran seven paired T-tests to recognize the top choice which has the minimum value for the ESC metric between all methods. Table 10 depicts the numerical values obtained from this test. As can be seen from these

results, the methods enjoy statistically different values of the ESC metric. The T-test's results indicate that using the DBSHD1 method leads to a remarkably lower value for the ESC metric with a *P*-value less than 0.05.

Table 10. Result of comparing different methods using the paired T-test

Others methods	Proposed methods	Difference	Confidence interval (0.95)	P-value
LR(17.71)	DBSHD2(0.14)	17.6	14.2, 21.00	P-value< 0.05
THR(15.91)	DBSHD2(0.14)	15.8	13.17, 18.43	P-value< 0.05
MAD(21.37)	DBSHD2(0.14)	21.26	15.9, 26.63	P-value< 0.05
IQR(14.88)	DBSHD2(0.14)	14.77	12.30, 17.24	P-value< 0.05
DES(4.15)	DBSHD2(0.14)	4.04	3.12, 4.96	P-value< 0.05
WMA(1.62)	DBSHD2(0.14)	1.51	1.01, 2.02	P-value< 0.05
REDMT(0.86)	DBSHD2(0.14)	0.76	0.39, 1.12	P-value< 0.05
DBSHD2(0.03)	DBSHD1(1.11)	0.30	0.011, 0.050	P-value< 0.05

By simulating the proposed methods and statistical analysis of their results, we observe that our methods prevent frequent migrations and thus, improve system efficiency. This occurs due to their insensitivity to the instantaneous explosions. Additionally, using the adaptive thresholds for efficient determining of the overloaded and underloaded hosts, the proposed methods significantly decrease the values of SLA violation and CR metrics. According to the results presented in Table 10, with regards to the ESC metric, there is a statistically significant difference between the proposed DBSHD1 and DBSHD2 methods and other benchmark methods. Based on these results, it can be concluded that the proposed DBSHD1 method outperforms the benchmark methods.

7. Conclusion and Future Work

In general, we proposed a proactive algorithm for detecting the source hosts which is insensitive to the workload explosions or bursts. The DBSHD method uses the mean load of the hosts, instead of their instantaneous load, in order to determine the source hosts. This job reduces the negative impact of the workload explosions. We attempted to maintain the weighted average load of the hosts in a dynamic range. This range and weights are calculated using the statistical analysis of the historical data of hosts. Further, a real-world dataset and simulator were employed to analyze the performance of the methods. The simulation results and statistical analysis for the proposed algorithms revealed that using the DBSHD method, the values of the parameters, like the migration counts, violations of the SLA, and CR were significantly reduced and a good trade-off was established between the performance and energy consumption, in comparison with the common methods.

The proposed methods are found efficient in terms of the ESC metric, compared to the benchmark algorithms. Also, the decrease in the values of SLA violation and CR were higher than the decrease in the amount of energy consumption. As future work, we can focus on the algorithm in order to reduce energy consumption while keeping the SLA violation at the same efficiency. The proposed algorithm aims to determine the overloaded and underloaded hosts in a manner that is insensitive to the fluctuations of the workloads or bursts. As a result, providing burst-aware solutions for other sub-problems of the VM consolidation process is subject to further investigations. Finally, proposing new methods for calculating the thresholds may improve the efficiency of the algorithm, since the efficiency of the algorithm is very dependent on the values of the thresholds.

Funding Not applicable.

Data Availability There is no any sensitive data collected or owned. Every data and materials are available for use.

Code Availability The experiment has been done in CloudSim simulator which is an open source.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no competing interest.

References

[1] M. H. Ferdaus, M. Murshed, R. N. Calheiros, and R. Buyya, "Multi-objective, Decentralized Dynamic Virtual Machine Consolidation using ACO Metaheuristic in Computing Clouds," *arXiv preprint arXiv:1706.06646,* 2017.

[2] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and Systems Management,* vol. 23, pp. 567-619, 2015.

[3] Y. Sharma, B. Javadi, W. Si, and D. Sun, "Reliability and energy efficiency in cloud computing systems: Survey and taxonomy," *Journal of Network and Computer Applications,* vol. 74, pp. 66-85, 2016.

[4] M. H. Ferdaus, "Multi-objective Virtual Machine Management in Cloud Data Centers," PhD thesis, Monash University, Melbourne, 2016.

[5] I. Pietri and R. Sakellariou, "Mapping virtual machines onto physical machines in cloud computing: A survey," *ACM Computing Surveys (CSUR),* vol. 49, p. 49, 2016.

[6] A. Beloglazov, "Energy-efficient management of virtual machines in data centers for cloud computing," Phd thesis , University of Melbourne, Department of Computing and Information Systems, 2013.

[7] Z. Li, C. Yan, X. Yu, and N. Yu, "Bayesian network-based virtual machines consolidation method," *Future Generation Computer Systems,* vol. 69, pp. 75-87, 2017.

[8] R. W. Ahmad, A. Gani, S. H. A. Hamid, M. Shiraz, A. Yousafzai, and F. Xia, "A survey on virtual machine migration and server consolidation frameworks for cloud data centers," *Journal of Network and Computer Applications,* vol. 52, pp. 11-25, 2015.

[9] G. Lovász, F. Niedermeier, and H. De Meer, "Performance tradeoffs of energy-aware virtual machine consolidation," *Cluster Computing,* vol. 16, pp. 481-496, 2013.

[10] 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," *Concurrency and Computation: Practice and Experience,* vol. 24, pp. 1397-1420, 2012.

[11] M. A. Khan, A. Paplinski, A. M. Khan, M. Murshed, and R. Buyya, "Dynamic Virtual Machine Consolidation Algorithms for Energy-Efficient Cloud Resource Management: A Review," in *Sustainable Cloud and Energy Services*, ed: Springer, 2018, pp. 135-165.

[12] A. Varasteh and M. Goudarzi, "Server consolidation techniques in virtualized data centers: A survey," *IEEE Systems Journal,* 2015.

[13] M. Arif, A. K. Kiani, and J. Qadir, "Machine learning based optimized live virtual machine migration over WAN links," *Telecommunication Systems,* vol. 64, pp. 245-257, 2017.

[14] B. R. B. A., "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers.," presented at the Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science, 2010.

[15] V. D. Reddy, G. Gangadharan, and G. S. V. Rao, "Energy-aware virtual machine allocation and selection in cloud data centers," *Soft Computing,* vol. 23, pp. 1917-1932, 2019.

[16] M. C. Silva Filho, C. C. Monteiro, P. R. Inácio, and M. M. Freire, "Approaches for optimizing virtual machine placement and migration in cloud environments: A survey," *Journal of Parallel and Distributed Computing,* vol. 111, pp. 222-250, 2018.

[17] S. Rahmani and V. Khajehvand, "Burst-aware virtual machine migration for improving performance in cloud," *International Journal of Communication Systems,* vol. 33, p. e4319, 2020.

[18] E. Arianyan, H. Taheri, and S. Sharifian, "Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers," *Computers & Electrical Engineering,* vol. 47, pp. 222-240, 2015.

[19] E. Arianyan, H. Taheri, and S. Sharifian, "Novel heuristics for consolidation of virtual machines in cloud data centers using multi-criteria resource management solutions," *The Journal of Supercomputing,* vol. 72, pp. 688-717, 2016.

[20] 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, pp. 755-768, 2012.

[21] P. H. Castro, V. L. Barreto, S. L. Corrêa, L. Z. Granville, and K. V. Cardoso, "A joint CPU-RAM energy efficient and SLA-compliant approach for cloud data centers," *Computer Networks,* vol. 94, pp. 1-13, 2016.

[22] A. A. El-Moursy, A. Abdelsamea, R. Kamran, and M. Saad, "Multi-Dimensional Regression Host Utilization algorithm (MDRHU) for Host Overload Detection in Cloud Computing," *Journal of Cloud Computing,* vol. 8, p. 8, 2019.

[23] F. Farahnakian, P. Liljeberg, and J. Plosila, "LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers," in *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on*, 2013, pp. 357-364.

[24] F. Farahnakian, T. Pahikkala, P. Liljeberg, and J. Plosila, "Energy aware consolidation algorithm based on k-nearest neighbor regression for cloud data centers," in *Utility and Cloud Computing (UCC), 2013 IEEE/ACM 6th International Conference on*, 2013, pp. 256-259.

[25] S. Y. Z. Fard, M. R. Ahmadi, and S. Adabi, "A dynamic VM consolidation technique for QoS and energy consumption in cloud environment," *The Journal of Supercomputing,* pp. 1-22, 2017.

[26] N. Hamdi and W. Chainbi, "A survey on energy aware VM consolidation strategies," *Sustainable Computing: Informatics and Systems,* vol. 23, pp. 80-87, 2019.

[27] M. Masdari, S. S. Nabavi, and V. Ahmadi, "An overview of virtual machine placement schemes in cloud computing," *Journal of Network and Computer Applications,* vol. 66, pp. 106-127, 2016.

[28] S. B. Melhem, A. Agarwal, N. Goel, and M. Zaman, "Markov prediction model for host load detection and VM placement in live migration," *IEEE Access,* vol. 6, pp. 7190-7205, 2018.

[29] R. Mohamadi Bahram Abadi, A. M. Rahmani, and S. H. Alizadeh, "Server consolidation techniques in virtualized data centers of cloud environments: A systematic literature review," *Software: Practice and Experience,* vol. 48, pp. 1688-1726, 2018.

[30] R. Zolfaghari and A. M. Rahmani, "Virtual machine consolidation in cloud computing systems: Challenges and future trends," *Wireless Personal Communications,* vol. 115, pp. 2289-2326, 2020.

[31] H. Li, G. Zhu, Y. Zhao, Y. Dai, and W. Tian, "Energy-efficient and QoS-aware model based resource consolidation in cloud data centers," *Cluster Computing,* vol. 20, pp. 2793-2803, 2017.

[32] S. Rahmani, V. Khajehvand, and M. Torabian, "Burstiness-aware virtual machine placement in cloud computing systems," *The Journal of Supercomputing,* vol. 76, pp. 362-387, 2020.

[33] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," in *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, Bangalore, India, 2010, p. 4.

[34] 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 & Electrical Engineering,* vol. 47, pp. 241-254, 2015.

[35] W. Zhu, Y. Zhuang, and L. Zhang, "A three-dimensional virtual resource scheduling method for energy saving in cloud computing," *Future Generation Computer Systems,* vol. 69, pp. 66-74, 2017.

[36] J.-p. Luo, X. Li, and M.-r. Chen, "Hybrid shuffled frog leaping algorithm for energy-efficient dynamic consolidation of virtual machines in cloud data centers," *Expert Systems with Applications,* vol. 41, pp. 5804-5816, 2014.

[37] H. M. Naeen, E. Zeinali, and A. T. Haghighat, "A stochastic process-based server consolidation approach for dynamic workloads in cloud data centers," *The Journal of Supercomputing,* vol. 76, pp. 1903-1930, 2020.

[38] A. A. Rahmanian, G. Dastghaibyfard, and H. Tahayori, "Penalty-aware and cost-efficient resource management in cloud data centers," *International Journal of Communication Systems,* vol. 30, p. e3179, 2017.

[39] R. Yadav, W. Zhang, O. Kaiwartya, P. R. Singh, I. A. Elgendy, and Y.-C. Tian, "Adaptive energy-aware algorithms for minimizing energy consumption and SLA violation in cloud computing," *IEEE Access,* vol. 6, pp. 55923-55936, 2018.

[40] M. H. Sayadnavard, A. T. Haghighat, and A. M. Rahmani, "A reliable energy-aware approach for dynamic virtual machine consolidation in cloud data centers," *The Journal of Supercomputing,* vol. 75, pp. 2126-2147, 2019.

[41] A. Aryania, H. S. Aghdasi, and L. M. Khanli, "Energy-Aware Virtual Machine Consolidation Algorithm Based on Ant Colony System," *Journal of Grid Computing,* pp. 1-15, 2018.

[42] S. Rahmani, V. Khajehvand, and M. Torabian, "Kullback-Leibler distance criterion consolidation in cloud," *Journal of Network and Computer Applications,* p. 102789, 2020.

[43] D. Alsadie, Z. Tari, and E. J. Alzahrani, "Online VM Consolidation in Cloud Environments," in *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, 2019, pp. 137-145.

[44] A. Abdelsamea, A. A. El-Moursy, E. E. Hemayed, and H. Eldeeb, "Virtual machine consolidation enhancement using hybrid regression algorithms," *Egyptian Informatics Journal,* 2017.

[45] Z. Yang, C. Li, A. Yun, and C. Liu, "A new trigger strategy based on live migration of the virtual machine," in *Control Engineering and Communication Technology (ICCECT), 2012 International Conference on*, 2012, pp. 677-680.

[46] 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,* vol. 7, pp. 9490-9500, 2019.

[47] A. Horri, M. S. Mozafari, and G. Dastghaibyfard, "Novel resource allocation algorithms to performance and energy efficiency in cloud computing," *The Journal of Supercomputing,* vol. 69, pp. 1445-1461, 2014.

[48] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience,* vol. 41, pp. 23-50, 2011.

[49] K. Park and V. S. Pai, "CoMon: a mostly-scalable monitoring system for PlanetLab," *ACM SIGOPS Operating Systems Review,* vol. 40, pp. 65-74, 2006.

[50] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomputing,* vol. 60, pp. 268-280, 2012.