IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 23, 2018, accepted September 23, 2018, date of publication October 2, 2018, date of current version October 25, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2872750

Adaptive Energy-Aware Algorithms for Minimizing Energy Consumption and SLA Violation in Cloud Computing

RAHUL YADAV[1], WEIZHE ZHANG[1], (Member, IEEE), OMPRAKASH KAIWARTYA[2], (Member, IEEE), PRABHAT RANJAN SINGH[3], IBRAHIM A. ELGENDY[1], AND YU-CHU TIAN[4], (Member, IEEE)

[1]School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China
[2]School of Science and Technology, Nottingham Trent University, Nottingham NG1 4FQ, U.K.
[3]School of Computer Science, Wuhan University of Technology, Wuhan 430070, China
[4]School of Electrical Engineering and Computer Science, Queensland University of Technology, Brisbane, QLD 4000, Australia

Corresponding author: Weizhe Zhang (wzzhang@hit.edu.cn)

This work was supported in part by the National Key Research and Development Plan under Grant 2016YFB0800801, in part by the National Science Foundation of China under Grants 61672186 and 61472108, and in part by the Australian Research Council under Grant DPDP170103305.

ABSTRACT In cloud computing, high energy consumption and service-level agreements (SLAs) violation are the challenging issues considering that the demand for computational power is growing rapidly, thereby requiring large-scale cloud data centers. Although, there are many existing energy-aware approaches focusing on minimizing energy consumption while ignoring the SLA violation at the time of a virtual machine (VM) selection from overloaded hosts. Also, they do not consider that the current network traffic causes performance degradation and thus may not really reduce SLA violation under a variety of workloads. In this context, this paper proposes three adaptive models, namely, gradient descent-based regression (Gdr), maximize correlation percentage (MCP), and bandwidth-aware selection policy (Bw), that can significantly minimize energy consumption and SLA violation. Energy-aware methods for overloaded host detection and VM selection from an overloaded host are necessary to improve the energy efficiency and SLA violation of a cloud data center after migrating all VM from underloaded host turn to idle host, which switch to energy-saving mode is also beneficial. Gdr and MCP are adaptive energy-aware algorithms based on the robust regression model, for overloaded host detection. A Bw dynamic VM selection policy selects VM according to the network traffic from the overloaded host under SLAs. Experimental results on the real workload traces show that the proposed algorithms reduce energy consumption while maintaining the required performance levels in a cloud data center using a CloudSim simulator to validate the proposed algorithms.

INDEX TERMS Cloud computing, cloud data center, energy-efficiency, green computing, host overloaded detection, regression method, service level agreements.

I. INTRODUCTION

Cloud computing has fundamentally transformed the way of traditional ownership model (computing capabilities are acquired in past) to current subscription model. It offers on-demand access to elastic resources as services with pay as you go model based on the actual usage of resources [1]. For fulfilling demand of computing resources requiring large-scale data centers. These data centers require an enormous amount of electric power to provide essential services to cloud users, and such consumption increases operating costs. The data centers consume approximately 1.3% of the total worldwide electricity supply; the rate is predicted to increase by 8% in 2020. If the essential steps are taken, cloud data center energy consumption can be minimize from the predicted worst case of 8000 TWh to 1200 TWh by the 2030 [2]. Unfortunately, large amounts of electric energy are wasted by servers during low workload. The servers resources utilization data collected from more than 5000 production servers over a six-month period show that most time servers operate at 10% to 50% of their full capacity, thereby leading to wasting energy during low utilization of resources [3].

An important question is when host become a overloaded host which against the SLA. The cloud services provider (CSP) is allocate virtual machine (VM) to cloud service users (CSU) according to their demand. Most of time CSU is not fully utilizing their given resources, thereby CSP think host still have some resources and allocate new VM to new CSU for decreasing total number active hosts. Meanwhile, previous CSU also demanding more resources at this time an overloading situation occur. Second, if the CSU wants to increase their existing VM resources on fully utilize host. So that time also overloading situation occurs.

In order to address host overloaded detection problem which is directly impact on energy consumption and SLA, we leverage the benefit of upper CPU utilization threshold for maximize utilization of host resources which leads small number of active hosts and switch rest of the hosts in energy-saving mode to minimize energy consumption [4]. Dynamic VM consolidation is a significant method for optimally utilizing computing resources from data centers. In this approach, the VMs are selected from the overloaded host and reallocated according to the current resource requirement to minimize the number of active hosts and service level agreements (SLAs) [5]. After migrating all VMs from under loaded hosts, thereby these idle hosts switched to an energy-saving mode with fast transition times to eliminate static energy and reduce total energy consumption [6]. The hosts will reactivate when workload demand increases. This method possesses has two main objectives: one is to achieve efficient energy consumption, and the other is to minimize the violation of service level agreements (SLAs). The ant colony optimization based VM Consolidation model consolidate VMs into a reduced number of active PMs according to the current resource requirements. However, the time complexity of ant colony optimization based model is high [7]. The regression-based utilization prediction model approximates not only the future CPU utilization but also memory utilization of VMs. However, this model is not a robust so sensitivity is high towards the outliers [6], [8].

In summary, most of the existing approaches consider the host CPU utilization threshold for host overloaded detection process and assume that the available bandwidth is always equal to base bandwidth thus leading high performance degradation. In addition, the previous works could not leverage the combination of energy consumption and SLA violations with network traffic. For example, when VM selected from overloaded host for reallocate or migrate to another appropriate host, the existing algorithms only consider VM size and VM current utilization for minimizing the energy consumption. But actually, SLA violation also affected by the network traffic.

In this context, this paper introduce regression-based algorithms called *Gdr* and *MCP* to set a dynamic upper CPU utilization threshold for detecting overloaded hosts. From these hosts, some VMs are migrated to another appropriate host to minimize performance degradation. A novel dynamic algorithm called Bandwidth-Aware Dynamic VM Selection Policy (*Bw*) is introduced to VM selection to balance the tradeoff among energy consumption, the number of migrations, the performance of hosts, and the total number of shutdown hosts. These algorithms have estimated the upper threshold for overloaded host detection and selection of VMs after host detected overloaded on the basis of statistical analysis of past CPU. The main contributions of this study are listed as follows:

- Adaptive heuristic algorithms *Gdr* and *MCP* are proposed to set a dynamic upper CPU utilization threshold for optimal utilization of host resources and detect overloaded hosts, thereby minimize the number of active hosts which means reducing the energy consumption as well as SLA violation. These algorithms have significantly minimized the number of reactivated hosts.
- A dynamic bandwidth-aware (*Bw*) VM selection algorithm introduced to Select VMs from an overloaded host. The basic idea of this algorithm is minimizing the VM migration time from overloaded host to appropriate host, thereby minimize overall SLA violation.
- The efficiency of algorithms *Gdr*, *MCP* and *Bw* is evaluated using the CloudSim simulator, and demonstrate its superiority by comparing to the several algorithms discuss in literature.

The rest of the paper organized as follows: In Section 2, we discusses previous works related to efficient-energy consumption in cloud data center; In Section 3, we presents the layer wise cloud Infrastructure as a Service (IaaS) architecture; The Section 4 is a main section of this paper, describes the algorithms for overloaded host detection and VM selection algorithm; In Section 5, we proposes an energy efficiency metric for measuring the effectiveness of the proposed algorithms in the cloud environment; In Section 6, introduces the simulation setup for the proposed algorithms; In Section 7, analyzes and compares the simulation results of the proposed algorithms; Finally, Section 8 elaborates the conclusions of the study and future research direction.

II. RELATED WORK

In this section, related literature on energy efficiency and SLA violation in cloud data center is reviewed. It is divided into three categories including energy-efficiency based on network, meta-heuristic approach, and adaptive threshold. We are discuss in following subsection.

A. ENERGY-EFFICIENCY BASED ON NETWORK

Energy-aware modeling has been extensively explored for cloud data centers. Detection of overloaded and underloaded hosts is also referred to as load balancing of servers. Most researchers have also focused on the network environment of the cloud data center. Schad *et al.* [9] experimented on Amazon EC2 and found that the 20% variation in performance is due to poor network connectivity. Ranganathan *et al.* [10] described a method for power management of servers at the collective system level instead of individual server level. This approach permits active servers to steal power from inactive servers. The problem of VM consolidation is considered NP-hard [11]. Therefore, the cost of finding the optimal solution in large-scale virtualized data centers (large number of hosts and VMs) is high. Esfandiarpoor *et al.* [12] proposed a VM consolidation algorithm for energy-aware application in cloud data center in consideration of structural features, such as racks and network topology. They also focused on the structure of the network and cooling system in a cloud data center that hosts the physical machines during VM consolidation. Specifically, racks and routers are employed without compromising the SLAs. In this way, low traffic routers or idle routers and cooling equipment can be turned off to minimize electricity consumption.

Dodonov and de Mello [13] introduced a method for grid computing to schedule distributed applications, and this method based on predictions of communication tasks. If the cost of migration is less than that of the predicted communication, then migration must be executed. This method works on grid computing but is inapplicable to a dynamic virtualized cloud data center because the cost of VM migration is higher than that of migration.

B. ENERGY-EFFICIENCY BASED ON META-HEURISTIC APPROACH

Meta-heuristic technique which has ability to find optimal path between given set of nodes with sink as destination [14]. Farahnakian *et al.* [7] introduced an online optimization meta-heuristic algorithm called Ant Colony System (ACS). This ACS-based VM Consolidation (ACS-VMC) approach uses artificial ants to consolidate VMs into a reduced number of active PMs according to the current resource requirements. These ants work in parallel to build VM migration plans based on a specified objective function. Ferdaus *et al.* [15] introduced an algorithm (AVVMC) based on Ant Colony Optimization(ACO) to the nonlinear resource allocation problem. The *AVVMC* seeks to find an optimal allocation of a limited amount of resources to the number of tasks to optimize their nonlinear objective function.

C. ENERGY-EFFICIENCY BASED ON ADAPTIVE THRESHOLD

Adaptive based algorithms focuses on the statistical analysis of past CPU utilization to determine an upper PU utilization threshold for detecting overloaded hosts. Elnozahy *et al.* [16] examined the energy management for resources in a single web-application platform with constant SLAs (response time) and the balancing of the load controlled by the application. Beloglazov *et al.* [17] proposed a cloud computing architectural framework to efficiently provision data center resources while meeting SLA requirements. They divided the VM consolidation problem into two parts: (1) the submission of new requests for VM provisioning and VM allocation on hosts and (2) the significant use of the current VM allocations. They used the MBFD algorithm to solve VM placement on hosts. This algorithm first sorts the current CPU utilization of all VMs in decreasing order and then allocates each VM to a host that provides an efficient energy consumption environment. Beloglazov and Buyya [8] introduced a adaptive based energy-aware approach. This approach focuses on the statistical analysis of CPU utilization history to determine an upper threshold for detecting overloaded hosts. However, placing a VM on the overloaded host causes performance degradation for this VM. As a solution, a VM controller should gather all information and provide an appropriate host during VM consolidation.

Wen *et al.* [18] introduced VM selection algorithm based on minimum migration policy. This policy selects the VMs, which deployed on the integrated hosts. To solve the target VM placement, they propose a VM placement algorithm based on an improved genetic algorithm. Using the encoding, crossover and mutation operations of the genetic algorithm, they obtain an effective solution for the VM placement problem. Zhu *et al.* [19] introduced a static CPU utilization threshold for detecting overloaded hosts. If the utilization of the host is more than 85% of its total capacity, then this host is detected as an overloaded host. This approach is unsuitable for dynamic workload because it cannot adapt to workload changes. Several current works are focusing on decision-making based on statistical analysis of historical data.

Bobroff *et al.* [20] introduced a dynamic server migration approach to improve the number of required resources and decrease the rate of SLA violation. This approach can forecast dynamic workloads over intervals less than the time scale of demand variability. The most current studies focus on managing thermal efficiency within cloud data centers. Sharma *et al.* [21] proposed a software-driven thermal management and temperature-aware workload placement to obtain additional energy savings. However, the performance of efficient-thermal management in cloud resources remains unexplored.

III. SYSTEM ARCHITECTURE

In a cloud computing platform, more than one heterogeneous VMs installed on a host. The virtualization technology provides administrative privileges to VM users within the guest operating system in that they can customize their run-time resources according to their specific needs [22], [23]. These VMs can run different types of the application simultaneously.

Each VM and host is characterized by parameters, such as CPU computing power defined in million instructions per second (MIPS), secondary storage provided by a network-attached storage or storage area network, primary storage (RAM), and network bandwidth [24]. In this architecture, the main objective of efficient energy consumption with the benefits of VMs can be achieved by consolidating the

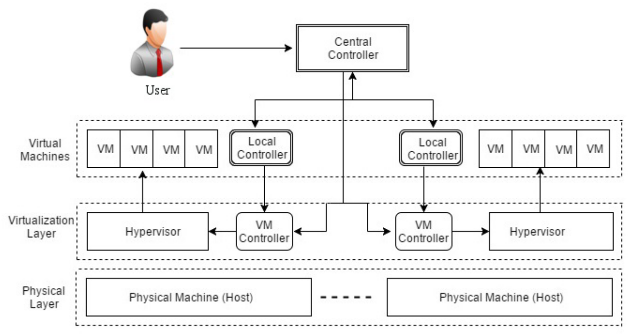

FIGURE 1. **Layer wise cloud IaaS architecture.**

computation load into a small number of servers while setting idle servers to an energy-saving mode [8], [15], [17]. The basic problem of energy consumption modeling is overloaded host detection from active hosts in the data center to match the current workload. This problem becomes challenging when we recognize the dynamic nature of data center in which VMs can request resources at any time and release the resources when their work is completed. Therefore, multiple heterogeneous VMs request resources and leave at any time. This cloud computing architecture can be considered a highly dynamic system. The regression based algorithms are especially useful in the dynamic nature of cloud architectures. This scenario allows us to recognize the performance of this complex system and is useful in designing data center resource management policies.

In Figure 1, the bottom layer of this architecture is a pool of physical servers (hosts) that consist of a large scale of computing power and storage resources. Above this layer is the virtualization layer, where users request the resources to run the heterogeneous application in the form of VMs and run their tasks with different SLAs. In this model, the three key players are central, local, and VM controllers. A local controller resides in each host as a separate VM and monitors the status of the VM and CPU utilization. The local controller should be able to decide when VM should be migrated from the host. The central controller resides in the single master host and gathers all information from the local controller to maintain the overall utilization of resources. The central controller decides on the optimal location of the VM. Finally, the VM controller resides in the hypervisor and is useful in resizing the VM and changing the power state of the host. The step-by-step workflow of the three controllers is described as follows:

- The local controller of each host continues to monitor the CPU utilization of the host and sets it into three specific fields, namely, overloaded host, underloaded host, and normal host. The three specific fields are categorized based on the following conditions.
  - The upT (12) of CPU utilization is determined by using *Gdr* and *MCP* overloaded host detection algorithm. If the current CPU utilization is greater than the estimated upT, then the host is overloaded. Local controller also selects the VMs from the overloaded host through the Bw VM selection algorithm.
  - After detecting the overloaded hosts, this system model will compare the current CPU utilization of remaining hosts and categorize the host with minimum CPU utilization as underloaded. If the migration of all VMs from an underloaded host is successful, then it will be set to sleep mode. Otherwise, it will remain active.
  - All remaining hosts are categorized into normal running hosts.
- The central controller continues to gather information from the local controller about the overall view of the resource utilization of the hosts and design the best optimal plan for VM placement through MBDF algorithm [17]. It issues the command to the VM controller for consolidation.
- The VM controller is useful in migrating and resizing the VMs.

A. ENERGY MODEL

Some components of the computing system (such as CPU, network, and memory) in the data center consume larger amounts of energy than do other components. Current studies show that the energy consumed by the processor of the host is directly propositional to the utilization of the processor. The utilization of the processor depends on the workload of the host and changes depending on the variability of the workload [25]. Therefore, the utilization of processor is a function of time. The overall energy consumption by the host can be defined as an integral function of power consumed by the host at a given period of time, and this function is described as follows [17]:

$$E = \int_{t_0}^{t_1} P(u(t))dt. \tag{1}$$

Where E represents the total electric energy consumed by the server. *P(u(t))* is a continuous function of the workload utilization at time t.

Moreover, we consider four different types of hosts namely, Fujitsu M1, Fujitsu M3, Hitachi TS10, and Hitachi SS10. The TABLE 2. is showing the features of these hosts. The energy consumption of the considered servers at different workload obtained from SPECpower [26]. The TABLE 1. is showing the energy consumption of these hosts at different workloads.

IV. ADAPTIVE BASED THRESHOLD FRAMEWORK FOR OVERLOADED HOST AND VM SELECTION

In this section, we present the proposed resource management algorithms along with its components for detecting overloaded hosts, and selection of VM from overloaded host to relocation to other hosts in the cloud datacenter in a holistic manner. The main notations and their meanings

TABLE 1. The electric energy consumed by the considered servers at different level of workload in watts(W).

Server	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Fujitsu M1	13.3	18.3	21.1	23.4	26.5	29.6	34.7	40.7	46.8	57.4	60
Fujitsu M3	12.4	16.7	19.4	21.4	23.4	26.1	29.7	34.8	41	47.1	51.2
Hitachi TS10	37	39.9	43.2	45.5	48.8	52.8	57.8	65.1	73.8	80.8	85.2
Hitachi SS10	36	38.8	41.2	43.7	46.3	49.4	53.1	58.8	64.2	67	69.7

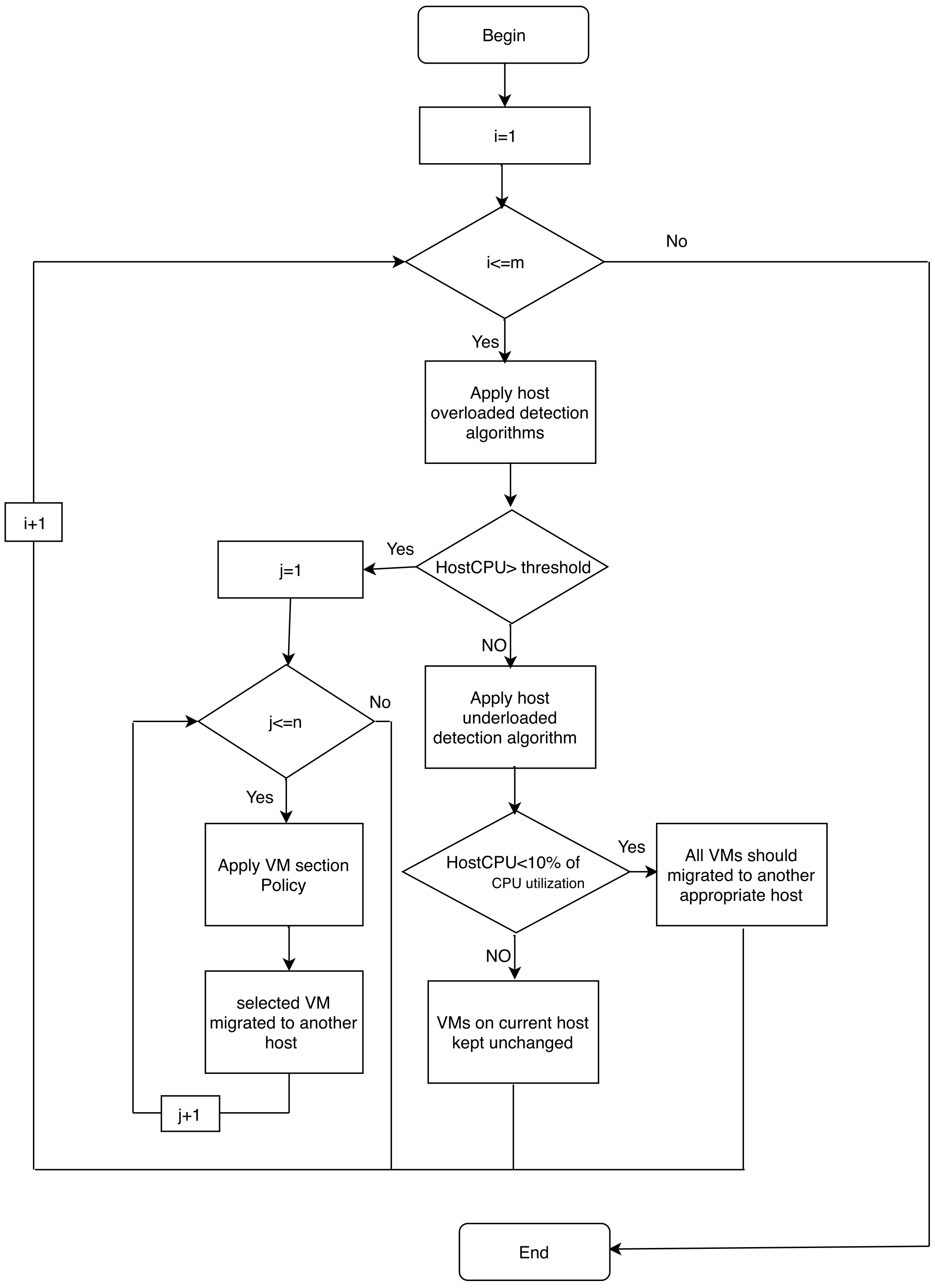
Begin
i=1
i<=m
No
Yes
Apply host overloaded detection algorithms
i+1
HostCPU> threshold
Yes
j=1
NO
j<=n
No
Apply host underloaded detection algorithm
Yes
Apply VM section Policy
HostCPU<10% of CPU utilization
Yes
All VMs should migrated to another appropriate host
NO
selected VM migrated to another host
VMs on current host kept unchanged
j+1
End

FIGURE 2. Adaptive threshold based framework for proposed algorithms.

used in throughout the paper. The FIGURE 2, is showing the proposed framework for resource management based on overloaded host detection algorithms (*Gdr* and *MCP*) and *Bw* for VM selection from overloaded hosts. In the first phase, the proposed algorithm lists the hosts, applies *Gdr* and *MCP* for overloaded host detection and compares the CPU utilization of hosts by using the dynamically estimated upper CPU utilization threshold. If the current CPU utilization of host exceeds the dynamically estimated value, then this host is predicted as an overloaded host. After detecting the overloaded host, all VMs of this host are identified and *Bw* algorithm applied to select the VM that needs to be migrated from the overloaded host. The algorithm *Bw* select a VM according to minimum migration time compare to other VMs on the same host. After the vmMigrateList is updated, the MBFD VM placement scheme [17] is actively applied on the selected VMs from the overloaded host and the VMs are allocated to the appropriate host. In the second phase, the method proposed in [17] is applied to find underloaded hosts. Thereafter, all the VMs from these hosts are migrated to the appropriate host and then turning these hosts into energy-saving mode. We discuss *Gdr*, *MCP* for overloaded host detection and *Bw* for VM selection in the following subsections.

A. GDR OVERLOADED HOST DETECTION

An adaptive heuristic based models provide a better result than that of traditional ones, such as static threshold based energy-aware techniques. We propose a heuristic-based upper CPU utilization threshold for detecting overloaded hosts and dynamic selection of VMs. The upper CPU utilization threshold estimated on the statistical analysis of past CPU utilization data [27]. The dynamic properties of the cloud environment are a huge concern for CSP. Therefore, the constant CPU utilization threshold of overloaded host detection is not an optimal solution for the dynamic workload of the cloud model. We proposed a novel algorithm, namely, *Gdr*, based on the machine learning technique for detecting the overloaded host. This algorithm calculates the upper CPU utilization threshold based on the historical dataset of CPU utilization, which is dynamically adjusted according to the historical workload of the CPU.

Gradient Descent is a popular first-order iterative optimization method. The cost function $\mathcal{J}(\theta_1, \theta_2)$ is minimized by an initial set of parameter values in the opposite direction of the gradient of the cost function $\nabla_{(\theta_1,\theta_2)}\mathcal{J}(\theta_1, \theta_2)$ with respect to θ_1, and θ_2 and then move toward a set of parameter values that minimize the cost function. The learning rate η determine the size of the steps that should be performed to reach a local minima [28]. However, in our approach does not have a local minima or maxima because the cost function $\mathcal{J}(\theta_1, \theta_2)$ is a convex function, which has a global minimum or maximum. Therefore, the proposed *Gdr* algorithm is the best means of finding the optimal solution for the cloud environment. The *Gdr* overloaded host detection algorithm is more efficient than the traditional approaches such as median absolute deviation (MAD), linear regression(LR), and ordinary least square(OLS). The model is calculated by using the following process: first, we must develop a LR model to understand the relationship between the value of input data X and output data Y.

$$Y_i = \theta_1 + \theta_2 X_i + \epsilon_i \tag{2}$$

$$\epsilon_i = Y_i - (\theta_1 + \theta_2 X_i) \tag{3}$$

Where, ϵ_i is independent variable called residuals. X_i and Y_i are the vector of past CPU utilization. The main objective of this model is to minimize the value of residuals ϵ_i. If the value of all residuals ϵ_i converges to the zero, then we found an optimal model in which all given data points are lie on the it. $i \in H$, where H is a set of m hosts CPU utilization in the cloud data center. The goal is to estimate the parameters, θ_1 and θ_2, which is usually called the intercept and slope of the fitted line in the given dataset respectively. To fit this line in given dataset by estimating the value of θ_1 and θ_2 that will minimize the sum of squared residuals describe as follows:

$$\min_{\theta_1,\theta_2} \mathcal{J}(\theta_1, \theta_2) = \frac{1}{2m}\sum_{i=1}^{m}\Big((\theta_1 + \theta_2 X_i) - Y_i\Big)^2 \tag{4}$$

For calculating the value of θ_1 and θ_2, we must to partial differentiate the equation (4) with respect to θ_1 and θ_2.

$$\frac{\partial \mathcal{J}(\theta_1, \theta_2)}{\partial \theta_1} = \frac{2}{2m}\sum_{i=1}^{m}\Big((\theta_1 + \theta_2 X_i) - Y_i\Big)$$

$$\hat{\theta_1} = \frac{1}{m}\sum_{i=1}^{m}\Big((\theta_1 + \theta_2 X_i) - Y_i\Big) \tag{5}$$

$$\frac{\partial \mathcal{J}(\theta_1, \theta_2)}{\partial \theta_2} = \frac{2}{2m}\sum_{i=1}^{m}\Big(((\theta_1 + \theta_2 X_i) - Y_i)X_i\Big) \tag{6}$$

$$\hat{\theta_2} = \frac{1}{m}\sum_{i=1}^{m}\Big(((\theta_1 + \theta_2 X_i) - Y_i)X_i\Big) \tag{7}$$

Second, we must update the value of $\hat{\theta_1}$ (5) and $\hat{\theta_2}$ (7) simultaneously until these values will converge to the optimal solution through the gradient descent method describe as follows:

$$\theta_1 = \theta_1 - \eta\hat{\theta_1} \tag{8}$$

$$\theta_2 = \theta_2 - \eta\hat{\theta_2} \tag{9}$$

Where, η is a learning rate to determine the size of steps to reach an optimal solution. If the value of η is large, then the oscillation of the model is fast and the model may fail to converge or even diverge to the optimal solution.If the value of η is very small then the convergence of the model to optimal solution is slow. Here we use the value of η is 0.001. For calculating the *upT*, we must first calculate the migration interval by using equation (10). Where, *MaxVmMigrationTime* represent the maximum VM migration time which is total depends on the size of the VM and bandwidth of the network within the data center and it's calculated using equation (18). *SchedulingInterval* represent the scheduling interval of the VM. Here, we use the value of *Scheduling-Interval* is 300, describe as follows:

$$MigrationIntervals = \frac{MaxVmMigrationTime}{SchedulingInterval} \tag{10}$$

$$Predicted = \theta_1 + \theta_2(n + MigrationIntervals) \tag{11}$$

Where, n represents the total number of VMs. *Predicted* (11) is an upper threshold without using a safety parameter. It is calculated by using gradient descent parameters θ_1 (8) and θ_2 (9). However, a safety parameter is the most important feature for *Gdr* host overloaded detection algorithm, which define how fast the system is consolidation the VMs.

$$upT = \rho * Predicted \tag{12}$$

Algorithm 1 Gdr Host Overloaded Detection

```
1: **Input:** Dataset of the CPU utilization
2: **Output:** Boolean // Host is overloaded or Not
3: for each i = 1 to m do
4:     $X_i \leftarrow$ CpuUtilHistory(i);
5:     $Y_i \leftarrow$ CpuUtilHistory(i+1);
6:     for each HostCPU $\in [H]$ do
7:         Calculate the $\theta_1$ and $\theta_2$ by using gradient descent.
8:         $\theta_1$ and $\theta_2 \leftarrow NULL$
9:         $\theta_1 \leftarrow \theta_1 - \frac{\eta}{m}\sum_{i=1}^{m}\Big((\theta_1 + \theta_2 X_i) - Y_i\Big)$
10:        $\theta_2 \leftarrow \theta_2 - \frac{\eta}{m}\sum_{i=1}^{m}\Big(((\theta_1 + \theta_2 X_i) - Y_i)X_i\Big)$
11:     end for
12: end for
13: MigrationIntervals $\leftarrow \frac{MaxVmMigrationTime}{SchedulingInterval}$
14: Predicted $\leftarrow \theta_1 + \theta_2\big(n + MigrationIntervals\big)$
15: $upT \leftarrow \rho \times$ Predicted
16: if Current utilization $> upT$ then
17:     return True;
18: else
19: return False;
20: end if
```

Where, ρ represents the safety parameter of *Gdr* algorithm. This parameter is changing according to the requirements of the system. If the value of ρ is small, which implies the less power consumption but high SLA violation rate or if the value is ρ is large, which implies a large amount of power consumption with low SLA violation. Here we take the value of $\rho = 1.8$. The pseudocode of *Gdr* host overloaded detection algorithm, which help in understanding the full workflow of the algorithm is discussed in **Algorithm 1**. Firstly, we obtain all data points of the VMs CPU utilization of the data center (line 3). For calculate the parameters θ_1 and θ_2 using gradient descent method. At the initial stage the value of the parameters θ_1 and θ_2 is Zero and it will change simultaneously in each iteration over m iterations (line 4-9). Afterwards, based on the current estimated value of θ_1 and θ_2, model will calculate the upper threshold (*upT*) using the equations (11) and (12) for detecting the overloaded hosts (lines 12-14). If the value of the upper threshold *upT* < current host CPU utilization then the host is categorized as overloaded host which required VM migration for reducing the SLA violation.

1) TIME COMPLEXITY ANALYSIS

The total time complexity is defined as O(initializing the vector x and y) + O(calculate the value of θ_1 and θ_2) + O(comparing current CPU utilization of all hosts with predicted upper threshold). To be specific, O(initializing the vector of x and y) is O(m); O(find the value of θ_1 and θ_2) is O(m×m); and O(comparing current CPU utilization of all hosts with predicted upper threshold) is O(m). Thus, total time complexity is $O(m^2)$.

B. MAXIMIZE CORRELATION PERCENTAGE (MCP) FOR OVERLOADED HOST DETECTION

In this section, we propose the second algorithm for choosing an adaptive upper CPU utilization threshold based on the correlation coefficient. The main concept of this algorithm is that the maximum percentage of the correlation between the utilization of resources by VMs running on an oversubscribed host, the higher probability of host overloading. According to this idea, we select maximum correlation percentage (MCP) of the hosts CPU utilization for setting an adaptive upper CPU utilization. To estimate the MCP, we used *partial regression coefficient* and *Pearson Correlation Coefficient)* [29], [30]. The *partial regression coefficient* used in the context of multiple linear regression analysis and give the quality of the estimation of the dependent variable. The *Pearson Correlation Coefficient* gives the proportion of variance of the predicting variable explained by the model.

Let $X_1, X_2, ..., X_m$ be m random variables representing the m number of hosts CPU utilization in the cloud data center. Let Y represent highly oversubscribe host among the m hosts. The matrix $\mathbf{X}$ is called an augmented matrix because the first column is composed only of 1.

$$X = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m-1,1} & \cdots & x_{m-1,m-1} \end{bmatrix} \quad and \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

A vector of predicted values of the dependent random variable $\hat{Y}$ is denoted by $\hat{y}$ and is obtained as follows.

$$\hat{y} = X\theta \quad and \ \theta = (X^T X)^{-1} X^T y \tag{13}$$

The regression square sum of the each observed value of y and its corresponding predicted value of y is obtained as follows.

$$SS_{\text{regression}} = \theta^T X^T y - \frac{1}{m-1}(1^T y)^2 \tag{14}$$

Where **1** is a row vector of 1's conformable with **y**. The total sum of squares of each observed value of y and its corresponding predicted value of y is obtained as follows.

$$SS_{\text{Total}} = y^T y - \frac{1}{m-1}(1^T y)^2 \tag{15}$$

The quality of CPU utilization prediction variable $\hat{Y}$ is evaluated as the proportion of the variance of the dependent variable explained by the independent variables. With this interpretation, the MCP define as follows.

$$MCP = \frac{1}{m}\sum_{i=1}^{m}\frac{SS_{\text{regression}}}{SS_{\text{total}}} \tag{16}$$

The pseudocode in **Algorithm 2** gets an $m\times 1$ vector H (line 1), which consist of a history of CPU utilization. At the initial stage, define **X** *m-1*×*m-1* augmented matrix consist of CPU utilization history. Afterward, calculate a matrix of θ by using equation (13). The value of $SS_{regression}$ determines that how far predicted model from the mean of the given data set. Calculate the $SS_{regression}$ by using equation (14). Then, In the same fashion, calculate SS_{Total} by using the equation (15). At last, we calculate *MCP* (line 17) by using the equation 16, which is helpful in finding the upper CPU utilization threshold. If the current CPU utilization of the host is greater than the calculated upper threshold, then this host is detected as an overloaded one and thus violates the SLAs.

Algorithm 2 Maximize Correlation Percentage (MCP)

```
1:  Input: H // History of Hosts CPU utilization
2:  Output: Boolean // Host is overloaded or not
3:  // Approximate the MCP value from regression function
    based on m hosts utilization history
4:  for i = 1 to m do
5:      for j = 1 to m do
6:          CpuUtil [i][j] ← ListCpuUtilHistory(j);
7:      end for
8:  end for
9:  end for
10: initialize augmented matrix X and vector y
11: // Calculated the value of $\theta$
12: $\theta = (X^T X)^{-1} X^T y$
13: // Calculated the value of $SS_{regression}$
14: $SS_{\text{regression}} = \theta^T X^T y - \frac{1}{m-1}(1^T y)^2$
15: // Calculated the value of $SS_{Total}$
16: $SS_{\text{Total}} = y^T y - \frac{1}{m-1}(1^T y)^2$
17: $MCP = \frac{1}{m}\sum_{i=1}^{m}\frac{SS_{\text{regression}}}{SS_{\text{total}}}$
18: $upT \leftarrow 1 - p(MCP)$
19: if $upT <$ Current utilization then
20:     return True;
21: else
22: return False;
23: end if
```

1) TIME COMPLEXITY ANALYSIS

The total time complexity is defined as O(find the value of θ) + O(initializing augmented matrix) + O(calculate $SS_{regression}$) + O(calculate SS_{Total}) + O(calculate *MCP*) + O(comparing current CPU utilization of all hosts with predicted upper threshold). To be specific, O(initializing augmented matrix) is O(m×m); O(find the value of θ) is O(m×m×m); O(calculate $SS_{regression}$) is O(m×m); O(calculate SS_{Total}) is O(m×m); O(calculate *MCP*) is O(m×m) and O(comparing current CPU utilization of all hosts with predicted upper threshold) is O(m). Thus, total time complexity is $O(m^3)$.

C. BANDWIDTH-AWARE DYNAMIC VM SELECTION POLICY (Bw)

Once it is decided that a host found overloaded, the next step is to select particular VM to migrate from this host. The VM migration is a costly operation that includes some amount of CPU processing on the source host, the link bandwidth between the source and destination hosts, the downtime of the services on the migrating VM, and the total migration time. The basic idea of this algorithm is that select a VM from an overloaded host whose current utilization and total migration time is minimum than others. In order to minimize the migration time of VM in dynamic VM allocation scheme, measurement of network utilization is an important aspect. We use the term bandwidth base B_{base} to refer to the estimated transfer rate, we used 1Gbps B_{base} for VM migration at any instant. In order to evaluate VM selection policy based on bandwidth measurements, we used available bandwidth as well as latency as follows: First, measure the total ping time; Second compute the available bandwidth; Third, compute the Bandwidth Transfer Component(BTC) of the transfer time by dividing the VM size (VM_{size}) and its current utilization ($VM_{\text{utilization}}$) by the available bandwidth (B_{base}); and finally, compute mean of total ping time (PT) with BTC for calculating *MigrationTime*. We describe the selection policy as follows:

$$V_{\text{i}} = \{v \mid v \text{ is the total number of VMs on the } i^{th} \text{ host}\}$$
$$BTC = \frac{VM_{\text{size}}(v) \times VM_{\text{utilization}}(v)}{B_{\text{base}}} \tag{17}$$
$$MigrationTime = BTC + \frac{1}{m}\sum_{i=1}^{m} PT_{\text{i}} \tag{18}$$

In the simulation of this selection policy, the migration time (*MigrationTime*) is calculated for all VM on an overloaded host and VM with the minimum migration time value is chosen.

The pseudocode in **Algorithm 3** gets a List of VM (line 1), which consist of past CPU utilization of all VMs on overloaded hosts. After the host detected overloaded, *Bw* algorithm calculated migration time (line 9) of all VMs and select minimum migration time VM compare from others VMs. At last, selected VM allocated to the appropriate host.

1) TIME COMPLEXITY ANALYSIS

The total time complexity is defined as O(calculate migration time for each VM using *Bw* algorithm) + O(sorting minimum migration time VM). To be specific, O(calculate migration time for each VM using *Bw* algorithm) is O(n); O(sorting minimum migration time VM) is O(n). Thus, time complexity

Algorithm 3 Bandwidth-Aware Dynamic VM Selection Policy

```
1: Input: VmList //list of VMs on overloaded host
2: output: Selected VM
3: vmToMigrate ← $NULL$
4: minMetric ← $MIN$
5: for each VM in vmList do
6:     $VM_{size}$ ← Total vmRam Size
7:     $VM_{utilization}$ ← Current Utilization of VM
8:     $BTC = \frac{VM_{\text{size}}(v) \times VM_{\text{utilization}}(v)}{B_{\text{base}}}$
9:     $MigrationTime = BTC + \frac{1}{m}\sum_{i=1}^{m} PT_{\text{i}}$
10:    if $MigrationTime >$ minMetric then
11:        minMetric ← metric
12:        vmToMigrate ← VM
13:    end if
14: end for
15: return vmToMigrate
```

is O(n). Where n is the number of VMs running on an overloaded host.

V. EFFICIENCY METRICS

To evaluate the effectiveness of the algorithms, their results are compared with other algorithms discussed in related works by using different metrics. The first metric is called the total electric energy consumed by the data center resources at different application service workloads, and the data are provided by the cloud service provider. The second metric is the average percentage of SLA violation introduced by Beloglazov and Buyya [8]. Such violation occurs when the provisioned VMs have not requested resources (or the average computing power of the shared host is not allocated to the requested VMs). This metric directly influences the level of quality of service, which is not negotiated between the cloud provider and users. If an SLA violation occurs, then the cloud service provider must pay penalty costs to users to compensate for the SLA violation.

A. TOTAL NUMBER OF HOST SHUTDOWNS

The excessive number of host deactivation and reactivation causes extra penalty for invoking the hosts back due to sudden increasing the demand of resources The proposed approaches switch underloaded or ideal host to energy-saving-mode to minimize the energy consumption. However, excessive shutdowns cause extra efforts and delay to revoke them when resources demand arises.

B. SLA VIOLATION METRIC

The value of SLA violations is vital for the VM placement algorithm. This metric was proposed in the work of Beloglazov *et al.* [17] to measure the SLAs delivered to VMs in cloud environment. and SLA violations metric could be defined as

$$ASLA = \frac{1}{m}\sum_{i=1}^{m}\frac{T^{i}_{overloaded}}{T^{i}_{total-active}} \times \frac{1}{n}\sum_{j=1}^{n}\frac{T^{j}_{pdt}}{T^{j}_{lta}} \tag{19}$$

where parameter m is the number of hosts in a data center; parameter $T^{i}_{overloaded}$ represents total time during which host detected overloaded. The parameter $T^{i}_{total-active}$ represents the total time of host i being in active state means it's provides service to their VMs. The papameter T^{j}_{pdt} is estimate of performance degradation for VM j, and T^{j}_{lta} is total CPU capacity requested by VM j during VMs' lifetime.

C. PERFORMANCE METRIC

The overall performance is maximized by electric energy consumption, average SLA violation, and the number of hosts reactivated from the energy-saving mode. If the requirement of computing power increases to manage current workload, energy saving-mode host switched on for allocating new workload. This process directly impacts the energy consumption of the data center. To solve this problem, we extended the performance metric introduced in by Beloglazov and Buyya [8], which described as follows:

$$Pertric = ASLA \times HS \times E \tag{20}$$

Where Pertric represents the overall performance metric, HS represents the total number of shutdown hosts after applying the algorithm, and E is the total electric energy consumption of the data center. $ASLA$ represents the average percentage of SLA violation in the data center.

D. NUMBER OF VM MIGRATION

The live migration of VMs is a expensive operation in terms of SLA, their are some factor directly impact on it: (1) the network traffic between sources and destination host; (2) application shutdown time over migrating VM; and (3) total migration time of the VM. The 10% of CPU utilization of host energy consumption is adding to SLA during all VM migrations in the hosts.

VI. SIMULATION SETUP

We focus on large-scale virtualized resources of cloud data center for providing virtual computing resources to users. This task is also called IaaS. However, deploying a real large-scale virtualized infrastructure is costly and brings the difficulty in conducting repeated experimental analysis and comparing the results of the proposed algorithm. Therefore, simulation is the best choice for evaluating the proposed algorithms for overloaded host detection and VM selection. In this study, we use the CloudSim toolkit [31] for analyzing and comparing the performance of the proposed *Gdr*, *MCP* for overloaded host detection and *Bw* for VM selection. The program is a modern open-source simulator and provides an IaaS cloud computing framework for repeatable experiments.

In our cloud computing simulation setup, we are installing 800 heterogeneous hosts. These hosts are Fujitsu M1,

TABLE 2. The characteristics of the hosts.

Server	CPU	Core	Clock Speed	Memory
Fujitsu M1	Xeon 1230	4	2.7 GHz	8 GB
Fujitsu M3	Xeon 1230	4	3.5 GHz	8 GB
Hitachi TS10	Xeon 1280	4	3.5 GHz	8 GB
Hitachi SS10	Xeon 1280	4	3.6 GHz	8 GB

TABLE 3. Amazon EC2 VM types.

VM Types	MIPS	Memory
Hight-CPU instance	2500	850 MB
Extra-Large instance	2000	3750 MB
Small instance	1000	1700 MB
Micro instance	500	613 MB

TABLE 4. The characteristics of workload dataset [8].

Date	N. of VMs	Mean(%)	SD(%)
03-03-2011	1052	12.31	17.09
06-03-2011	898	11.44	16.83
09-03-2011	1061	10.7	15.57
22-03-2011	1516	9.26	12.78
25-03-2011	1078	10.56	14.14
03-04-2011	1463	12.39	16.55
09-04-2011	1358	11.12	15.09
11-04-2011	1233	11.56	15.07
12-04-2011	1054	11.54	15.15
20-04-2011	1033	10.43	15.21

Fujitsu M3, Hitachi TS10, and Hitachi SS10. The TABLE 2. is showing the features of these servers. The TABLE 1. is showing the electric energy consumptions of these servers at the different workloads.

The CPU clock speeds of a server term as MIPS. The cores of Fujitsu M1, Fujitsu M3, Hitachi TS10, and Hitachi SS10 server are mapped as 2700, 3500, 3500, and 3600 MIPS. The network bandwidth of each server is 1 Gbps. The TABLE 3. is showing Amazon EC2 VM types characteristics.

Simulation-based experiments must be conducted using real workload trace of data center servers, which are applicable to real cloud environments. For this purpose, we use the real workload data provided by PlanetLab as part of the CoMon project [32].

More than a thousand heterogeneous VM CPU utilization data from more than 500 heterogeneous servers worldwide used. The TABLE 4. is showing the features of a given dataset for each day.

VII. SIMULATION RESULT AND ANALYSIS

To evaluate the performance of *Gdr*, *MCP* for overloaded host detection and *Bw* for VM selection, we use real-time CPU utilization data of heterogeneous servers described in TABLE 3. The proposed algorithms are simulated, and their results are analyzed and compared with other algorithms discussed in for overloaded host detection and VM selection. The compared algorithms for overloaded host detection are median absolute deviation (MAD), linear regression (LR), and inter-quartile range (IQR); for comparison of

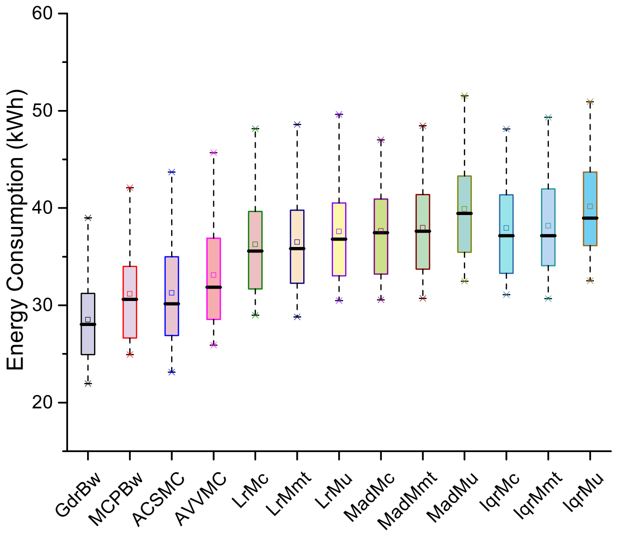

FIGURE 3. The comparison of the simulation results at the hourly measurement of power consumption of the data center.

VM selection algorithms, minimum migration time (MMT), maximum correlation (MC), and minimum utilization (MU) is used [8]. We also compared proposed algorithms with evolutionary algorithms based on ant colony optimization (ACO) such as *ACSMC* and *AVVMC* [7], [15]. We discuss the simulation results and their comparison in the following subsections.

A. ENERGY CONSUMPTION

We analyze and compare the simulation results of the proposed algorithms with *MAD*, *LR*, *IQR* in [8] and *ACSMC* and *AVVMC* [7], [15]. The experiment is executed for 10 days of workload data that are shown in TABLE 4. The simulation results are presented in FIGURE 3. Notably, energy consumption significantly reduces using the proposed algorithms for overloaded host detection (*Gdr*, *MCP*) with VM selection policy compared with techniques used in [7], [8], and [15]. The variation in energy consumption with the proposed algorithms is from 23 kWh to 43 kWh and the median is 30 kWh. Therefore, the data center consumes 30 kWh on most days. The average median of the results of other algorithms is 37 kWh. Therefore, the energy consumption result of the proposed algorithms is 12% less than those of the other algorithms. In other words, the overall energy consumption of the data center is significantly reduced by *Gdr*, *MCP*, and *Bw*. The energy consumption of the data center by using the techniques in [7], [8], and [15] is greater than the median of *GdrBw* and *MCPBw*.

B. AVERAGE SLA VIOLATION

The SLA violation is a big concern in user prospective. As shown in FIGURE 4, the average percentage of SLA violation of the proposed algorithms is significantly less than that of the combined techniques. The average median of SLA violation results of the combined algorithms is 8.9%. Therefore, the average percentage of SLA violation of the

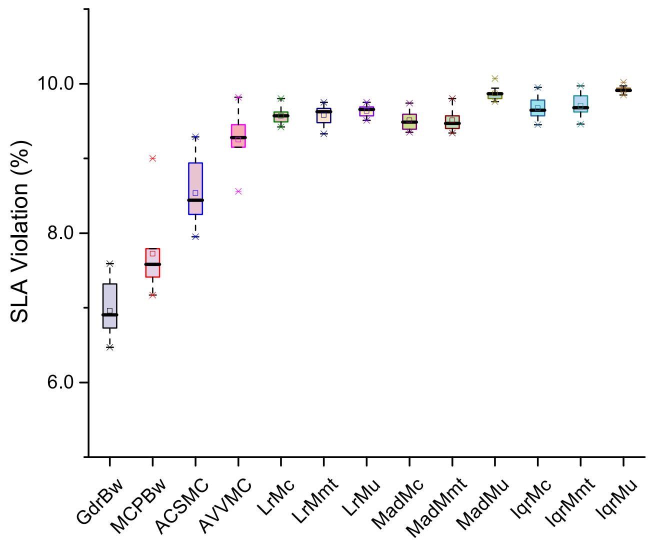

FIGURE 4. The Comparison of the simulation results, average SLA violation percentage.

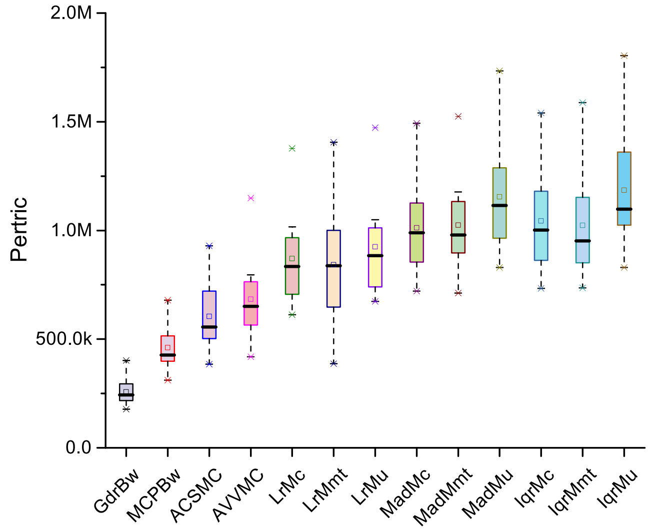

FIGURE 5. The comparison of the simulation results of a performance metric.

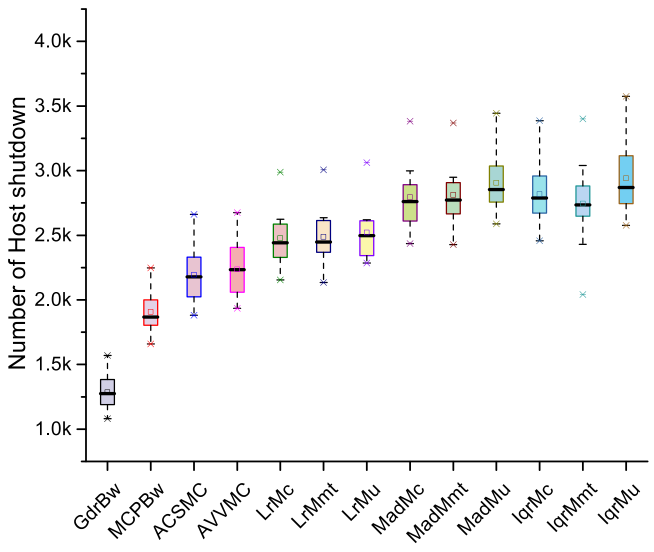

FIGURE 6. The comparison of the simulation results of the number of the hosts shutdown.

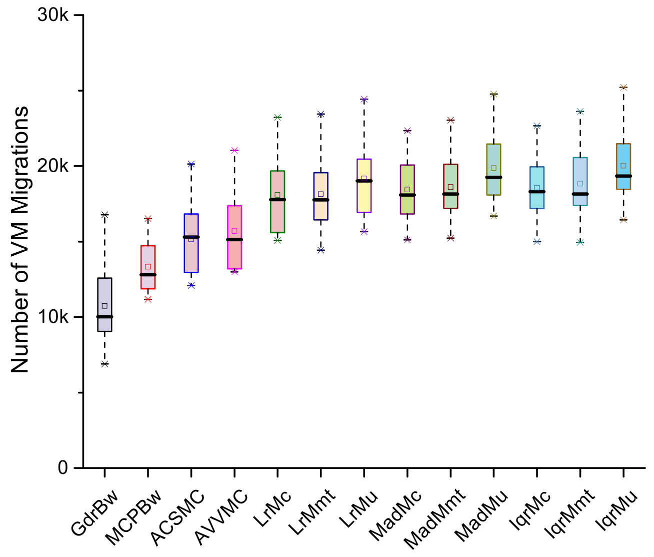

FIGURE 7. The comparison of the simulation results of the number of the VM migration.

proposed algorithms is 6.9%, which is 22% less than that of the combined algorithms.

C. *Pertric* (PERFORMANCE METRIC)

We discuss the overall performance of the cloud data center by using *Pertric*. The cloud provider aims to maximize the overall performance with minimized electric energy consumption, average SLA violation, and the number of reactivated hosts. FIGURE 5 shows the results of the performance metric by using the proposed algorithms.

The performance of the proposed algorithms is better than that of previous algorithms in terms of *Pertric*.

D. NUMBER OF HOSTS SHUTDOWN AND VMs MIGRATION

The simulation results of the number of shutdown hosts and VM migrations are analyzed and compared. If the number of reactivated hosts increases, as a result, the energy consumption of the data center also increases. It means the hosts are reactivated for allocating new VMs and shutdown after migrating all VMs from underloaded hosts.

We collect the data on the number of hosts turning to energy saving mode by implementing the proposed algorithms during simulation. As shown in FIGURE 6, the approximate median of the *Gdr* algorithm is 1300 hosts, and this value is much lesser than that of previous algorithms. The simulation results indicate that the proposed algorithms work better than does other algorithms because of the few numbers of reactivated hosts. Few numbers of reactivated hosts are directly proportional to the energy consumption of the data center.

TABLE 5. Summary of the one-way anova test.

Source of Variation	df	SS($\times10^{10}$)	MS($\times10^{10}$)	F ratio	p-Value	F critical
Between Groups	12	833.8.5	69.4	12.5	1.88E-15	1.85
Within Groups	104	575.1	5.53			
Total	116	1409.03				

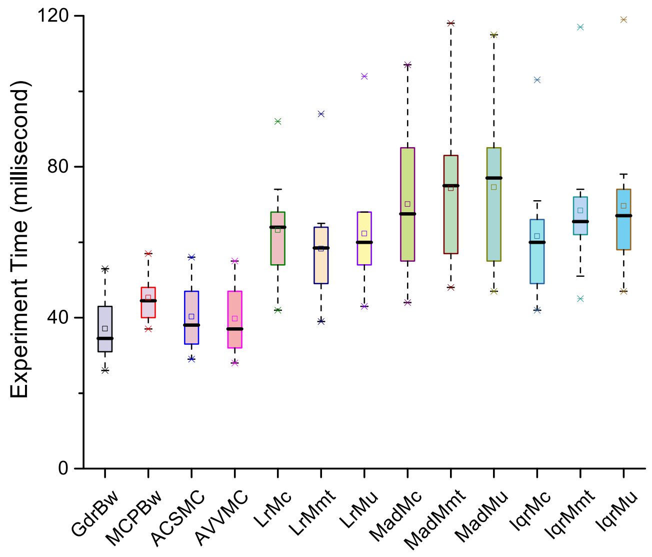

FIGURE 8. The comparison of the simulation results of the experiment time.

In the experimental environment, we use only 800 hosts. However, the number of shutdown hosts is high because of reactivation of hosts. The FIGURE 6. shows that using the proposed algorithms minimizes 22% of host reactivation compared with traditional algorithms. The simulated data on the number of VM migrations caused by the proposed algorithms are obtained. The FIGURE 7. shows that the number of VM migrations is directly proportional to the performance degradation of VMs during the active state of CPU utilization. The high number of VM migrations directly influences the SLAs, and such effect is unfavorable for the cloud provider and users. The reason is that the cloud provider must be pay penalty costs to users in such a situation. Performance degradation is also undesirable for users.

E. EXPERIMENT TIME

The experiment time determines as total execution time taken by the particular algorithm in the given data set. The FIGURE 8 shows that how fast algorithm executed on a given cloud environment. The proposed algorithms execute must faster than other algorithms discussed in literature work. The median execution time of *GdrBw* is 34 millisecond, which is must lesser than other algorithms.

F. STATISTICAL ANALYSIS

Statistical analysis validated the proposed algorithm, and the results demonstrated the efficiency of the proposed algorithm compared with other approaches. One-way ANOVA on the Pertric Matrices is conducted to analyze the trade-off between minimizing the overall energy consumption and maximizing the QoS of the data center demonstrated in TABLE 5. Where df, SS, and MS represents the degree of freedom, the sum of squares, and mean square respectively. Based on the One-way ANOVA result, *GdrBw* significantly reduced energy consumption as well as maximized QoS, compared with *MadMc, MadMmt, MadMu, IqrMc, IqrMmt, LrMc, LrMmt, LrMu, ACSMC, AVVMC*, and *IqrMu*. TABLE 5 shows that the F ratio (12.5) is greater than the F critical value (1.85), which indicates that the null hypothesis is rejected and the population means are significantly different from one another at the 0.05 level. Therefore, the *GdrBw* algorithm is significantly different from other algorithms, such as *MadMc, MadMmt, MadMu, IqrMc, IqrMmt, IqrMu, LrMmt, LrMc* and *LrMu* with p-value of 1.88E-15.

VIII. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, energy-aware algorithms proposed to improve the energy efficiency and minimize the SLA violation in cloud environment. The simulation results show that: (1) regarding the energy efficiency, the *Gdr* host overloaded detection algorithm improving energy consumption better than the *MCP* algorithm; (2) during the VM selection from overloaded host considering CPU, memory, and network traffic factor is more effective than a single factor such as CPU. Moreover, the algorithms proposed in this paper are more effective than the other energy-aware algorithms regardless the workload types. In future, We are planning to propose Thermal-aware algorithm for VM placement. Further, we analyze how much energy efficiency, SLA violation improvements and reduction in the operation cost is obtained.

REFERENCES

[1] R. Yadav and W. Zhang, "MeReg: Managing energy-SLA tradeoff for green mobile cloud computing," *Wireless Commun. Mobile Comput.*, vol. 2017, Nov. 2017, Art. no. 6741972.
[2] A. S. G. Andrae and T. Edler, "On global electricity usage of communication technology: Trends to 2030," *Challenges*, vol. 6, no. 1, pp. 117–157, 2015.
[3] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 12, pp. 33–37, Dec. 2007.
[4] J. K. Verma *et al.*, "Enabling green computing in cloud environments: Network virtualization approach toward 5G support," *Trans. Emerg. Telecommun. Technol.*, p. e3434, May 2018.
[5] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *Proc. INFOCOM*, Apr. 2011, pp. 71–75.
[6] M. A. Khoshkholghi, M. N. Derahman, A. Abdullah, S. Subramaniam, and M. Othman, "Energy-efficient algorithms for dynamic virtual machine consolidation in cloud data centers," *IEEE Access*, vol. 5, pp. 10709–10722, 2017.

[7] F. Farahnakian *et al.*, "Using ant colony system to consolidate VMs for green cloud computing," *IEEE Trans. Services Comput.*, vol. 8, no. 2, pp. 187–198, Mar./Apr. 2015.

[8] 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 Comput., Pract. Exp.*, vol. 24, no. 13, pp. 1397–1420, Sep. 2012.

[9] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, "Runtime measurements in the cloud: Observing, analyzing, and reducing variance," *Proc. VLDB Endowment*, vol. 3, nos. 1–2, pp. 460–471, 2010.

[10] P. Ranganathan, P. Leech, D. Irwin, and J. Chase, "Ensemble-level power management for dense blade servers," *ACM SIGARCH Comput. Archit. News*, vol. 34, no. 2, pp. 66–77, 2006.

[11] E. Feller, C. Morin, and A. Esnault, "A case for fully decentralized dynamic VM consolidation in clouds," in *Proc. 4th IEEE Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2012, pp. 26–33.

[12] S. Esfandiarpoor, A. Pahlavan, and M. Goudarzi, "Structure-aware online virtual machine consolidation for datacenter energy improvement in cloud computing," *Comput. Elect. Eng.*, vol. 42, pp. 74–89, Feb. 2015.

[13] E. Dodonov and R. F. de Mello, "A novel approach for distributed application scheduling based on prediction of communication events," *Future Gener. Comput. Syst.*, vol. 26, no. 5, pp. 740–752, 2010.

[14] P. R. Singh, M. A. Elaziz, and S. Xiong, "Modified spider monkey optimization based on Nelder–Mead method for global optimization," *Expert Syst. Appl.*, vol. 110, pp. 264–289, Nov. 2018.

[15] M. H. Ferdaus, M. Murshed, R. N. Calheiros, and R. Buyya, "Virtual machine consolidation in cloud data centers using ACO metaheuristic," in *Proc. Eur. Conf. Parallel Process.* Cham, Switzerland: Springer, 2014, pp. 306–317.

[16] E. N. M. Elnozahy, M. Kistler, and R. Rajamony, "Energy-efficient server clusters," in *Proc. Int. Workshop Power-Aware Comput. Syst.* Berlin, Germany: Springer, 2002, pp. 179–197.

[17] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, 2012.

[18] Y. Wen, Z. Li, S. Jin, C. Lin, and Z. Liu, "Energy-efficient virtual resource dynamic integration method in cloud computing," *IEEE Access*, vol. 5, pp. 12214–12223, 2017.

[19] X. Zhu *et al.*, "1000 Islands: Integrated capacity and workload management for the next generation data center," in *Proc. Int. Conf. Autonomic Comput. (ICAC)*, Jun. 2008, pp. 172–181.

[20] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing SLA violations," in *Proc. 10th IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, May 2007, pp. 119–128.

[21] R. K. Sharma, C. E. Bash, C. D. Patel, R. J. Friedrich, and J. S. Chase, "Balance of power: Dynamic thermal management for Internet data centers," *IEEE Internet Comput.*, vol. 9, no. 1, pp. 42–49, Jan. 2005.

[22] A. Aliyu *et al.*, "Cloud computing in VANETs: Architecture, taxonomy, and challenges," *IETE Tech. Rev.*, pp. 1–25, 2017.

[23] A. Aliyu, H. Abdullah, O. Kaiwartya, M. J. Usman, S. O. A. Rahman, and A. Khatri, "Mobile cloud computing energy-aware task offloading (MCC: ETO)," in *Proc. ICCCS*, p. 359, 2017.

[24] I. Elgendy, W. Zhang, C. Liu, and C.-H. Hsu, "An efficient and secured framework for mobile cloud computing," *IEEE Trans. Cloud Comput.*, to be published.

[25] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," in *Proc. Int. Conf. Autonomic Comput. (ICAC)*, Jun. 2008, pp. 3–12.

[26] *All Published Specpowerssj2008 Results*. Accessed: May 12, 2017. [Online]. Available: https://www.spec.org/ $power_ssj2008/results/power\_ssj2008.html$

[27] R. Yadav, W. Zhang, H. Chen, and T. Guo, "MuMs: Energy-aware VM selection scheme for cloud data center," in *Proc. 28th Int. Workshop Database Expert Syst. Appl.*, Aug. 2017, pp. 132–136.

[28] S. Ruder. (2016). "An overview of gradient descent optimization algorithms." [Online]. Available: https://arxiv.org/abs/1609.04747

[29] H. Abdi, "Partial regression coefficients," *The SAGE Encyclopedia of Social Science Research Methods*. Thousand Oaks, CA, USA: SAGE, 2004.

[30] J. Riu and F. X. Rius, "Univariate regression models with errors in both axes," *J. Chemometrics*, vol. 9, no. 5, pp. 343–362, 1995.

[31] R. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exp.*, vol. 41, no. 1, pp. 23–50, 2011.

[32] K. Park and V. S. Pai, "CoMon: A mostly-scalable monitoring system for PlanetLab," *ACM SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 65–74, 2006.

RAHUL YADAV received the M.S. degree in computer science from the Department of Computer Science and Mathematics, South Asian University, New Delhi, India, in 2015. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Harbin Institute of Technology, China. Since 2015, he has been a Doctoral Candidate. He has authored conference and journal papers. He is actively involved in the research on efficient-energy management, cloud computing, fog computing, optimal utilization of data center resources, machine learning, and cost-efficient virtual machine consolidation.

WEIZHE ZHANG (M'06) is currently a Professor with the School of Computer Science and Technology, Harbin Institute of Technology, China. He has authored over 100 academic papers in journals, books, and conference proceedings. His research interests are primarily in parallel computing, distributed computing, cloud and grid computing, and computer network. He serves on a number of journal editorial boards. He has edited more than 10 international journal special issues as a guest editor and has served for many international conferences as the chair or a committee member.

OMPRAKASH KAIWARTYA received the Ph.D. degree in computer science from Jawaharlal Nehru University, New Delhi, India, in 2015. He was a Research Associate with Northumbria University, Newcastle upon Tyne, U.K., and a Post-Doctoral Research Fellow with Universiti Teknologi Malaysia. He is currently a Lecturer with the School of Science and Technology, Nottingham Trent University, U.K. His research interest focuses on Internet of connected vehicles, electronic vehicles charging management planning and recommendation, and Internet-of-Things use cases implementation in wireless sensor networks. He is an Associate Editor of SCI journals, including the *IET Intelligent Transport Systems*, the *EURASIP Journal on Wireless Communication and Networking*, the IEEE ACCESS, and *Ad-Hoc and Sensor Wireless Networks*. He is also a Guest Editor of many recent special issues in reputed journals, including the IEEE INTERNET OF THINGS JOURNALS, the IEEE ACCESS, and the *MDPI Sensors*.

PRABHAT RANJAN SINGH received the M.Sc. degree in computer science from South Asian University, India. He is currently pursuing the Ph.D. degree with the School of Computer Science, Wuhan University of Technology, China. He focuses on the application of load balancing in cloud computing. His research interests include multi-objective optimization, swarm intelligence, and evolutionary algorithms.

IBRAHIM A. ELGENDY received the M.Sc. degree from the Computer Science Department, Faculty of Computers and Information, Menoufia University, Egypt, in 2016. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Harbin Institute of Technology, China. He has been a Demonstrator and an Assistant Lecturer with the Faculty of Computers and Information, Menoufia University, since 2012. His research interests include cloud computing, mobile-edge computing, and distributed computing.

YU-CHU TIAN (M'09) received the Ph.D. degree in computer and software engineering from The University of Sydney, Sydney, NSW, Australia, in 2009, and the Ph.D. degree in industrial automation from Zhejiang University, Hangzhou, China, in 1993. He is currently a Professor with the School of Electrical Engineering and Computer Science, Queensland University of Technology, Brisbane, QLD, Australia. His research interests include big data computing, distributed computing, cloud computing, real-time computing, computer networks, and control systems.