IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 30, 2017, accepted May 2, 2017, date of publication June 1, 2017, date of current version June 27, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2711043

Energy-Efficient Algorithms for Dynamic Virtual Machine Consolidation in Cloud Data Centers

MOHAMMAD ALI KHOSHKHOLGHI, MOHD NOOR DERAHMAN, AZIZOL ABDULLAH, SHAMALA SUBRAMANIAM, AND MOHAMED OTHMAN

Faculty of Computer Science and Information Technology, University Putra Malaysia, Serdang 43400, Malaysia

Corresponding author: Mohammad Ali Khoshkholghi (khosh.kholghi63@gmail.com)

ABSTRACT Cloud computing has become a significant research area in large-scale computing, because it can share globally distributed resources. Cloud computing has evolved with the development of large-scale data centers, including thousands of servers around the world. However, cloud data centers consume vast amounts of electrical energy, contributing to high-operational costs, and carbon dioxide emissions. Dynamic consolidation of virtual machines (VMs) using live migration and putting idle nodes in sleep mode allows cloud providers to optimize resource utilization and reduce energy consumption. However, aggressive VM consolidation may degrade the performance. Therefore, an energy-performance tradeoff between providing high-quality service to customers and reducing power consumption is desired. In this paper, several novel algorithms are proposed for the dynamic consolidation of VMs in cloud data centers. The aim is to improve the utilization of computing resources and reduce energy consumption under SLA constraints regarding CPU, RAM, and bandwidth. The efficiency of the proposed algorithms is validated by conducting extensive simulations. The results of the evaluation clearly show that the proposed algorithms significantly reduce energy consumption while providing a high level of commitment to the SLA. Based on the proposed algorithms, energy consumption can be reduced by up to 28%, and SLA can be improved up to 87% when compared with the benchmark algorithms.

INDEX TERMS Cloud computing, energy efficiency, service level agreement, virtual machine consolidation, data center.

I. INTRODUCTION

In recent years, cloud computing has become popular because of its ability to offer utility-oriented IT services over the Internet to global users. Cloud computing is a paradigm to develop scalable on-demand virtualized resources based on a pay-as-you-go model [1]. Different types of applications, from scientific to business, can utilize cloud-based services in various forms, including software, hardware, and data. The biggest IT companies, such as Google, Amazon, Microsoft, and IBM, have developed their cloud data centers around the world to support cloud services. Cloud data centers ideally allocate resources to users in a way that satisfies the required Quality of Service (QoS) determined by the cloud subscribers through the Service Level Agreement (SLA). In cloud computing, an SLA is defined as a two-sided contract between the cloud provider and its users, and it determines the content of services provided, level of performance, prices, and penalties for not providing the services. Any breach of the QoS leads to SLA violation, and consequently, a penalty must be paid by service providers [2].

Because of the rapid growth of cloud services and their corresponding technologies, cloud infrastructures have become more complicated and complex. Hence, resource management is one of the most prominent issues in modern cloud environments, directly affecting the efficient deployment of cloud services. Modern data centers provide a high level of performance and optimization; however, a new concern is energy consumption. Total electricity use by data centers in 2010 was estimated to be approximately 1.5% of all electricity consumption in the world and this number has increased to 3% in 2016 [3]. Google's data centers consumed 260 million Watts of electricity in 2013, which is enough to consistently power 200,000 homes. On the other hand, on average, 30% of cloud servers exploit 10-15% of their resource capacity most of the time. Therefore, energy-efficient resource management can be addressed to decrease both operational costs and environmental impacts. VM consolidation is one of the most productive techniques in energy-efficient resource management in cloud computing; this technique enhances resource utilization and decreases energy consumption.

Consolidation refers to the live migration of VMs between hosts with little in the way of performance interruption. The aim of consolidation is moving the VMs to a minimal number of hosts and switching the idle hosts to power saving modes [4]. Aggressive consolidation of VMs to minimize energy consumption may lead to performance degradation so that the system cannot deliver the expected quality of service, consequently leading to an increase in SLA violations [5]. Hence, the consolidation mechanism should keep possible SLA violations low while decreasing energy consumption as soon as possible.

In this paper, we propose a dynamic and adaptive energy-efficient VM consolidation mechanism considering SLA constraints for cloud data centers. The main contributions of the paper are as follows:

- Develop an overloading host detection algorithm using an iterative weighted linear regression method to determine two utilization thresholds and avoid performance degradation.
- Develop a power and SLA-aware VM selection algorithm using three different policies to select adequate VMs that need to be migrated to other hosts.
- Develop a two-phase VM placement algorithm that can be utilized for the effective placement of new VMs and the VMs selected for consolidation.
- Develop an underloading host detection algorithm using a vector magnitude squared of multiple resources to consolidate active hosts and switch them to a power saving mode.
- Develop a distributed energy-efficient dynamic VM consolidation mechanism by employing the proposed algorithms according to the utilization of multiple host resources.
- Conduct extensive simulation and performance analysis of the proposed mechanism.

The rest of this paper is organized as follows. Section 2 discusses related works. Section 3 introduces the system model presented in this study. The proposed VM consolidation mechanism is discussed in section 4. Section 5 presents a performance evaluation of the proposed mechanism. A summary and future works are given in section 6.

II. RELATED WORKS

Verma *et al.* [6] designed pMapper, a power-aware workload placement controller used for heterogeneous virtualized server clusters. This controller takes into account power consumption, SLA requirements, and migration costs. pMapper consists of a performance manager, migration manager, and power manager along with an arbitrator. Based on the information provided by these managers, the arbitrator determines the required size of the VMs. pMapper uses DVFS, server power switching and VM consolidation as the power management strategies. The authors proposed a power and migration cost tradeoff, a first fit decreasing and, a minimum power packing as three possible placement algorithms. As an advantage, these algorithms can be used for different types of workloads.

Beloglazov *et al.* [7] proposed a consolidation mechanism that sets two fixed thresholds regarding CPU utilization. In the case that CPU utilization increases more than the upper threshold or drops lower than the lower threshold, some VMs will be selected and migrated over to other hosts. For this reason, the authors performed an experiment to determine the best set of upper and lower thresholds to reduce power consumption while keeping SLA violations low. Also, the authors suggested using three VM selection policies. The first policy is Minimization of Migration (MM). The MM policy consolidates the least number of VMs to other hosts so that CPU utilization goes below the upper threshold. The Random Choice policy (RC) selects the VMs randomly. The Highest Potential Growth policy (HPG) selects the VMs that have the lowest CPU utilization relative to their total required CPU capacity. The results showed that the MM policy with a lower threshold set to 30% and an upper threshold set to 70% provides the best results. This study showed good results in both energy consumption and SLA violation rate; however, it has two weaknesses. First, the power consumption model only takes into account CPU power usage. Second, the consolidation mechanism is static because the thresholds are defined as fixed values, and this issue reduces the scalability of the approach for various workloads. However, our proposed mechanism considers the power consumption of all the server's components.

Because static thresholds are not adequate for unpredictable and dynamic workloads, Beloglazov and Buyya [8] proposed a dynamic VM consolidation mechanism for reducing both energy consumption and SLA violations in cloud data centers. This mechanism uses historical data of resource utilization for determining the adaptive thresholds for each server. The Median Absolute Deviation (MAD), Local Regression (LR) and Interquartile Range (IQR) policies are introduced for determining the dynamic upper thresholds. The LR policy is based on the Loess method and aims to determine the upper threshold by finding a regression curve that approximates future data. Moreover, the authors presented two VM selection policies. The Minimum Migration Time (MMT) policy selects the VMs with the least time needed for migration, and the Maximum Correlation policy (MC) selects a VM that have the highest correlation of the CPU utilization with other VMs. This mechanism takes into account the power consumption of all server components; however, the CPU is the only considered factor for consolidating the VMs, which is a weakness.

Mhedheb *et al.* [9] proposed a load and thermal-aware VM consolidation for cloud data centers. The approach aims to balance both load and temperature so that the system is not encountered with overutilization or high temperature while reducing the energy consumed by the servers. For this reason, the authors proposed Thermal-aware Scheduler (ThaS). ThaS utilizes DVFS as the power management technique,

schedules VMs regarding CPU temperature and sends VMs to the hosts with the least CPU usage and temperature.

Taheri and Zamanifar [10] introduced a two-phase VM consolidation mechanism to cope with the problem of incomplete migrations. Perplex VMs are the VMs that should be consolidated but that have no place in other hosts. Therefore, the system terminates the migration and replaces the VMs to the prior place. This issue leads to a waste of CPU capacity and power and increases the network's overhead. Based on the proposed framework, in the first phase, VMs from the over-utilized hosts migrate to other hosts and then in the second phase, VMs from underutilized hosts are sent to other hosts.

Some studies have used bin-packing optimization problem to propose VM placement policies. Shi *et al.* [11] proposed both offline and online VM placement algorithms by modifying first fit bin-packing algorithms. Keller *et al.* [12] proposed several VM placement policies that leveraged the different combinations of VM and server sorting methods. The results revealed that sorting the VMs in increasing order and sorting the servers in decreasing order of CPU utilization can improve the energy usage rather than other policies. Farahnakian *et al.* [13] introduced a machine learning-based dynamic VM consolidation technique to reduce the number of active hosts while optimizing resource utilization in the data centers. The proposed technique uses a reinforcement learning method to learn the optimal power mode to an agent; this is accomplished using past information and then switching off the idle nodes.

Another study has proposed an energy-efficient scheduling for high-performance computing jobs in virtualized clusters [14]. This solution used both DVFS and server consolidation techniques to reduce energy while optimizing the acceptance ratio job. Chawarut and Woraphon [15] proposed a CPU reallocation algorithm by combining of DVFS and VM live migration techniques for energy efficiency in real time services. This algorithm selects the VMs that need to be migrated in terms of the lowest CPU utilization, longest completion time and highest CPU utilization.

Lim *et al.* [16] proposed a power-aware technique (called PADD) to reduce energy consumption using live VM migrations considering SLA expectations in data centers. To this aim, PADD leverages two levels of buffering to cope with workload variations: a local buffer and global buffer. The local buffer refers to 10% of the CPU capacity reserved in each server. The global buffer is a reserved pool of CPU capacity across all the servers. Using these buffers, PADD minimizes the number of migrations and avoids SLA violations by allocating the reserved CPU capacity to any increased demands in the case of rapid variation of incoming workload. If the local buffer is less than a specified level, some VMs will be migrated to other servers. However, this solution defines local and global buffers in terms of CPU capacity and ignores other resources.

Wang *et al.* [17] proposed a distributed live VM migration mechanism in cloud data centers. The proposed mechanism uses load vectors by which each server collects information about the incoming workload from other servers. The load vector maintains the source index, destination index, and the amount of the source's CPU utilization. This information is needed for migration decisions. The proposed mechanism leverages a double-threshold technique in terms of CPU utilization to make the decision for VM migrations. In an experiment, the authors evaluated the different values of the lower and the upper thresholds, and the results showed that a lower threshold at 10% and upper threshold at 90% obtains the best results in terms of reducing both energy consumption and SLA violation. The study presents three scenarios for selecting which VMs to migrate. First, a VM where the current utilization goes lower than the upper threshold can be chosen. Second, several VMs can be selected in case there is no VM that can satisfy the first scenario. In the last scenario, one or some of VMs utilize a vast amount of CPU capacity so that they can overload the destination server; in this case, the proposed mechanism ignores them. However, the mechanism is static because the thresholds are defined as fixed values, and this issue makes the mechanism inappropriate for various types of workloads.

Zhou *et al.* [18] proposed an Adaptive Three-threshold Energy-aware (ATEA) VM placement algorithm to reduce both energy and SLA violation in cloud data centers. ATEA classifies the servers into hosts with little load, hosts with a light load, hosts with a moderate load, and hosts with high load. This classification is based on three thresholds: T_l, T_m and T_h ($0 \leq T_l \leq T_m \leq T_h \leq 1$), where T_l, T_m, T_h and U denote the lower threshold, median threshold, higher threshold, and CPU utilization, respectively. ATEA consolidates VMs only from hosts with little load ($0 \leq U \leq T_l$) and high load ($T_h \leq U$). These thresholds are defined using two mathematical techniques: K-means Clustering Algorithm-Median Absolute Deviation (KAM) and K-means Clustering Algorithm-Average Interquartile Range (KAI). The results showed that KAM obtained better results than KAI in both energy and SLA metrics. A tabular comparison of the related works is presented in Appendix I.

III. THE SYSTEM MODEL

The cloud system presented in this study consists of a large-scale data center included many heterogeneous physical servers. The cloud is an IaaS environment distributed worldwide. CPU performance, network bandwidth and the amount of RAM characterize each server. CPU performance is defined in Millions of Instructions Per Second (MIPS). The system storage is Network Attached Storage (NAS) which is common in clouds because NAS enables the distribution of VM live migration. The system has no knowledge about the application workload arriving into the system. In other words, the system is knowledge-free, and the proposed mechanism is workload-independent. The application requests are submitted to the system by multiple users, and their requirements are provided by one or several heterogeneous VMs. Cloud applications have a broad range of workload types, from

High-Performance Computing (HPC) to web-applications. The Cloud Service Provider (CSP) makes an SLA contract with consumers upon required QoS, and must pay a penalty if there is an SLA violation. As shown in Fig. 1, consumers send their requests to the global manager in charge of brokering the new demands and managing the VM migration to the available hosts. Each host has a local manager responsible for monitoring and managing the host resources. In fact, the VM consolidation algorithms are implemented in this module. The local broker monitors the host resources and makes decisions based on the available resources. Virtual Machine Manager (VMM) coordinates which VMs are to be started, switched to sleep mode, and shut down.

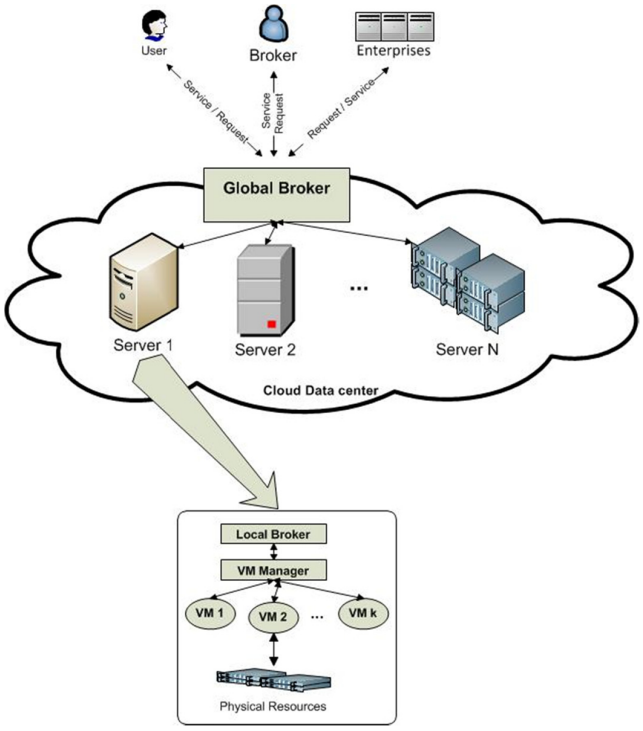

FIGURE 1. The system model.

IV. THE PROPOSED VM CONSOLIDATION MECHANISM

Using VM consolidation, cloud providers can optimize their resource utilization while reducing power consumption of data centers. Our proposed VM consolidation mechanism includes four algorithms, as follows:

- Overloading host detection: Distinguishes when hosts should be considered overloaded, in which case one or several VMs are reallocated to other hosts to reduce host utilization.
- Underloading host detection: Distinguishes when hosts should be considered underloaded, in which case all the VMs are consolidated to other hosts; then, the host is switched to the sleep mode.
- VM selection: Chooses the most suitable VMs to be migrated from overloaded hosts.
- VM placement: Discovers the most suitable destination host for the selected VMs.

Fig. 2 shows the overall diagram of our VM consolidation mechanism.

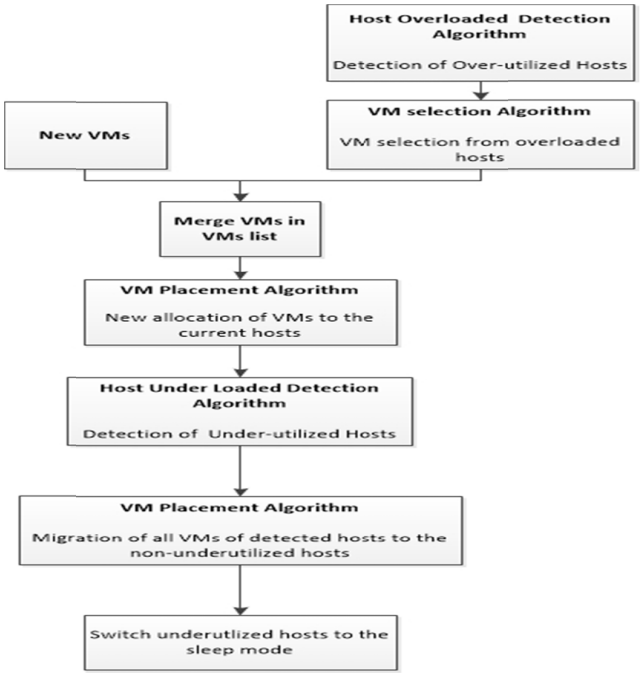

FIGURE 2. Overall diagram of the proposed VM consolidation mechanism.

A. OVERLOADING HOST DETECTION ALGORITHM

The objective of the overloading host detection algorithm is to recognize when a host is overloaded. Each host executes this algorithm periodically. Detection is based on the usage of host resources which are CPU, RAM, and bandwidth. Whenever the algorithm is invoked, it initiates the IWLR algorithm to dynamically determine the utilization thresholds for each of the three resources. In the event a host is overloaded, one or several VMs are selected for migration to other hosts, hence bringing the utilization under acceptable thresholds. Because the proposed consolidation mechanism is an adaptive mechanism for different types of workload, an adaptive method to detect overloaded hosts is proposed.

1) ITERATIVE WEIGHTED LINEAR REGRESSION (IWLR)

Regression is a statistical method for quantitative data analysis that is used to predict the future values of data. Regression is widely used for predictions in various fields [19]. Regression can be used in two models: simple regression in the case of one input and multiple regression for more than one input. The target of regression is approximating a regression function (linear or non-linear), which estimates the relationship between input variable X and output variable Y by the regression line. The proposed algorithm uses a simple weighted linear regression to predict future host utilizations.

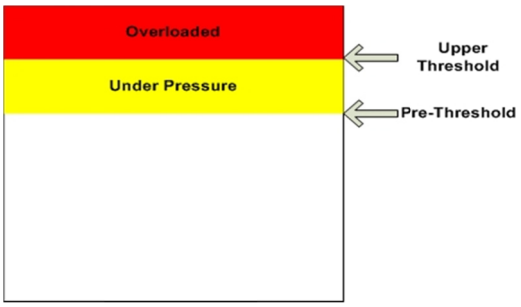

FIGURE 3. Host utilization thresholds.

The simple regression line is shown in (1).

$$Y = \beta_0 + \beta_1 X \tag{1}$$

Where Y is the dependent variable and X is the independent variable. β_0 and β_1 are regression coefficients and are derived from the least squares technique [20] as follows:

$$\widehat{\beta_0} = \bar{Y} - \widehat{\beta_1}\bar{X} \tag{2}$$

$$\widehat{\beta_1} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n}(X_i - X)^2} \tag{3}$$

Where $\bar{X}$ and $\bar{Y}$ are the means of X and Y observations, and $\widehat{\beta_0}$ and $\widehat{\beta_1}$ are estimations of β_0 and β_1 , respectively. For each observation (x_i, y_i), a neighborhood weight is assigned using the tricube weight function presented in [21] and [22] as follows:

$$T(u) = \begin{cases} (1 - |u|^3)^3 & \text{if } |u| < 1 \\ 0 & |u| > 1 \end{cases} \tag{4}$$

Based on the above formula, the neighborhood weight is defined as follows:

$$w_i(x) = T\left(\frac{x_n - x_i}{x_n - x_1}\right) = (1 - (\frac{x_n - x_i}{x_n - x_1})^3)^3 \tag{5}$$

Where x_i and x_n are i^{th} and the last observations, respectively. IWLR uses K iterations to detect K future values of the host utilization. For n data values (previous host utilization), the regression line is defined as follows:

$$\begin{aligned} \hat{y}_1 &= \beta_0 + \beta_1 x_n \\ \hat{y}_2 &= \beta_0 + \beta_1 \hat{y}_1 \\ &\cdots\cdots \\ \hat{y}_k &= \beta_0 + \beta_1 \hat{y}_{k-1} \end{aligned} \tag{6}$$

As shown in Fig. 3, IWLR determines two thresholds: the upper threshold and pre-threshold. Given a value of k, if the future host utilization is predicted to be higher than the total capacity (100%) in the next value ($i = 1$), the host will be marked as an overloaded host. However, when IWLR detects in other future values ($i = 2$ to $i = k$) that the host utilization is above the total capacity (100%), $\hat{y}_1$ is determined as the pre-threshold, and the host is marked as under pressure; in this case the host does not accept any new VMs.

In the proposed algorithm, we set $k = 2$ which means IWLR predicts two future values.

$$\begin{aligned} \hat{y}_1 &= \beta_0 + \beta_1 x_n \\ \hat{y}_2 &= \beta_0 + \beta_1 \hat{y}_1 \end{aligned} \tag{7}$$

In this case, there is the following:

$$\begin{cases} x_n \text{ is upper threshold}, & \text{if } c \cdot \hat{y}_1 \geq 1 \\ x_n \text{ is pre threshold}, & \text{if } c \cdot \hat{y}_2 \geq 1 \end{cases} \tag{8}$$

Where c is the intensity constant. Algorithm 1 and Algorithm 2 show the overloading host detection algorithm based on the IWLR method.

Algorithm 1 Overloading Host Detection Algorithm

Input: *host*
Output: *overloaded detection*
(1) *UTC ← IWLR(CPU).upperThreshold;*
(2) *PUC ← IWLR(CPU).utilPrediction;*
(3) *UTM ← IWLR(Memory).upperThreshold;*
(4) *PUM ← IWLR(Memory). utilPrediction;*
(5) *UTB ← IWLR(BW).upperThreshold;*
(6) *PUB ← IWLR(BW). utilPrediction;*
(7) ***if*** *((PUC or PUM or PUB)>=1)* ***then***
(8) *underPressureList ← host;*
(9) *Host will not accept new VM;*
(10) ***else***
(11) ***if*** *((UTC or UTM or UTB)>=1)* ***then***
(12) *overloadedList ← host;*
(13) ***end if***
(14) ***end if***
(15) ***return*** *underPressureList;*
(16) ***return*** *overloadedList;*

B. VM SELECTION ALGORITHM

As described in the preceding section, as the first part of consolidation mechanism, all the overloaded hosts are detected. Then, one or several VMs will be selected from each detected host using VM selection algorithm so that host utilization drops below the threshold. This algorithm is iterative and after selecting each VM, the utilizations of the host resources are checked again. In the case the host is still overloaded, more VMs will be selected. Three policies are proposed for this algorithm in this section. The algorithm is shown in Algorithm 3.

1) MAXIMUM POWER REDUCTION POLICY

The Maximum Power Reduction (MPR) policy selects and migrates a VM v that reduces the host power consumption after migration more than other VMs allocated to the host. Let VM_j be a set of VMs allocated to the host i, then the MPR

Algorithm 2 IWLR Algorithm

Input: *host utilization*
Output: *upperThreshold, utilPrediction*
(1) ***for*** $i = 1$ *to* n ***do***
(2) $x_i \leftarrow i$;
(3) $y_i \leftarrow$ *utilHistory(i);*
(4) $w_i \leftarrow$ *calculate using equation (5);*
(5) $x_i \leftarrow x_i{}^* w_i$;
(6) $y_i \leftarrow y_i{}^* w_i$;
(7) ***end for***
(8) *calculate* β_0, *using equation (2);*
(9) *calculate* β_1, *using equation (3);*
(10) $utilPrediction = \beta_0 + \beta_1{}^* currentUtil(h)$;
(11) $upperThreshold = utilPrediction$;
(12) *update* x, y *and* w;
(13) *update* β_0 *and* β_1;
(14) ***for*** $i = 2$ *to* k ***do***
(15) $KpredictUtil(i) = \beta_0 + \beta_1{}^* utilPrediction$;
(16) $utilPrediction \leftarrow KpredictUtil(i)$;
(17) ***end for***
(18) ***return*** *upperThreshold;*
(19) ***return*** *utilPrediction;*

Algorithm 3 VM Selection Algorithm

Input: *overloadedHostList, hostVMlist*
Output: *selectedVMList*
(1) ***foreach*** *host in overloadedHosList* ***do***
(2) ***foreach*** *VM in hostVMlist* ***do***
(3) *selectedVM←NULL;*
(4) *proposed VM selection technique;*
(5) *selectedVMlist←selectedVM;*
(6) ***end for***
(7) *currentCPUutil←currentCPUutil - selectedVMCPUutil;*
(8) *currentRAMutil←currentRAMutil - selectedVMRAMutil;*
(9) *currentBWutil←currentBWutil - selectedVMBWutil;*
(10) ***if***((*currentCPUutil* < *upperThreshold*) && (*currentRAMutil* < *upperThreshold*) && (*currentBWutil* < *upperThreshold*)) ***then***
(11) *break;*
(12) ***else***
(13) $hostVMlist \leftarrow hostVMlist - selectedVM$;
(14) *go to line 2;*
(15) ***end if***
(16) ***end for***
(17) ***return*** *selectedVMList;*

policy tries to find a set $V \in VM_j$ defined in (9).

$$V = \begin{cases} \left\{ \begin{array}{l} L|L \in VM_j, u_i - \sum_{v \in L} u(v) < T_{up}, |L| \rightarrow min, \\ P|u(v)| \rightarrow \max \end{array} \right\}, & \text{if } u_i > T_{up} \\ \varnothing, & otherwise \end{cases} \quad (9)$$

Where u_i is the utilization of the host i, T_{up} is the upper threshold, u (v) is the fraction of CPU utilization allocated to v and P |u(v)| is the power consumed by v in host i. The MPR policy is shown in Algorithm 4.

Algorithm 4 Maximum Power Reduction

(1) ***foreach*** *VM in hostVMlist* ***do***
(2) *selectedVM←NULL;*
(3) *maxPower←MIN;*
(4) *power←power(host, VM);*
(5) ***if*** *power* > *maxpower* ***then***
(6) *selectedVM←VM;*
(7) *selectedVMlist←selectedVM;*
(8) *maxpower←power;*
(9) ***end if***
(10) ***end for***

2) TIME AND POWER TRADEOFF POLICY

The Time and Power Tradeoff (TPT) policy selects and migrates a VM v that has the best trade-off between the least migration time and the most power reduction after migration relative to the other VMs allocated to the host. Let VM_j be a set of VMs allocated to the host i, then the TPT policy tries to find a set $V \in VM_j$ defined in (10).

$$V = \begin{cases} \left\{ \begin{array}{l} L|L \in VM_j, u_i - \sum_{v \in L} u(v) < T_{up}, |L| \rightarrow min, \\ \{P|u(v)| \rightarrow \max \,\&\, t(v) \rightarrow min\} \end{array} \right\}, & \text{if } u_i > T_{up} \\ \varnothing, & otherwise \end{cases} \quad (10)$$

Where u_i is the utilization of the host i, T_{up} is the upper threshold, $u(v)$ is the fraction of CPU utilization allocated to v, $P|u(v)|$ is the power consumed by v in the host i, and $t(v)$ is the migration time of v defined in (11).

$$Migration\ time = \frac{RAM(v)}{BW_i} \quad (11)$$

3) VIOLATED MIPS-VMs POLICY

The Violated Mips-VMs (V-VMs) policy uses a different technique to select the VMs that will be migrated. V-VMs selects all the VMs in the host that encounter a CPU Mips violation. In other words, a VM v that its allocated Mips is less than the requested Mips will be selected and migrated to the other host. Let VM_j be a set of VMs allocated to the host i, then the V-VMs policy finds a set $V \in VM_j$ defined in (12).

$$V = \begin{cases} \left\{ \begin{array}{l} L|L \in VM_j, u_i - \sum_{v \in L} u(v) < T_{up}, \\ \frac{u_a(v)}{u_r(v)} < 1 \end{array} \right\}, & \text{if } u_i > T_{up} \\ \varnothing, & otherwise \end{cases} \quad (12)$$

Where u_i is the utilization of the host i, T_{up} is the upper threshold, $u(v)$ is the fraction of CPU utilization allocated

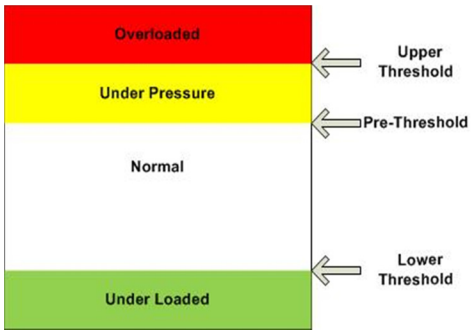

FIGURE 4. Different host utilization regions.

to v, $u_a(v)$ is the allocated Mips to the v and $u_r(v)$ is the requested Mips.

C. VM PLACEMENT ALGORITHM

After detecting the overloaded hosts and selecting the adequate VMs, then in this step, VMs are required to place to the best destination hosts. The target is finding the hosts where the selected VMs can run at a minimum energy and are least likely to commit SLA violations. For this reason, the Best RAM and Bandwidth placement algorithm (BRB) is proposed.

1) BEST RAM AND BANDWIDTH PLACEMENT ALGORITHM (BRB)

BRB is invoked in two phases: first, to place the selected VMs migrated from the overloaded hosts and second, to place the VMs migrated from the underloaded hosts. In the first phase, all the selected VMs are sorted in decreasing order according to their amount of RAM. Then from the top of the list, the VMs are checked against the hosts to find the best one available. As shown in Fig. 4, the hosts are categorized into four types based on their utilization: overloaded, under pressure, underloaded and normal. Normal hosts refer to hosts which their utilizations for three resources are upper than low threshold and lower than pre-threshold. First, BRB investigates the hosts in the normal list. If there are any hosts which have enough CPU, RAM, and BW for the VM, and if they are not overloaded after placement, then Mark is calculated using (13) for those hosts; finally, the host with a minimum Mark is chosen as the destination.

$$Mark = \frac{RAM(VM)}{available\ BW(host)} \tag{13}$$

In the case there is no host that can be selected from the normal list, BRB investigates the hosts on the underloaded list and repeats the procedure. If there is no adequate host on the underloaded list as well, then a new host is launched. This procedure is performed for all selected VMs. Algorithm 5 shows the BRB algorithm in the first phase.

After placing all the selected VMs in the first phase, and detecting underloaded hosts, then selected VMs migrated from the underloaded hosts are placed in the second phase. The first part of BRB in the second phase is the same as the first phase. First, BRB checks normal hosts, as described before in the first phase.

However, the difference is when the algorithm cannot find the destination among normal hosts. In this case, the underloaded hosts are divided into two lists: receivedVMlist, which refers to those hosts that received any VM in the first phase, and otherHostslist, which have not received any VM in the first phase. The reason is BRB tries to switch off under loaded hosts as much as possible, but increasing the VM migrations increases SLA violations due to interruption. Therefore, BRB tries to decrease VM migration in the second phase. For this aim, BRB first investigates the hosts on receivedVMlist. In the case that there is no adequate host to be selected, then otherHostslist is investigated. In the end, if there is still no selected host, a new host will be launched. Algorithm 6 shows the algorithm of BRB in the second phase. This algorithm can be used not only for VM consolidation but also as a resource allocation and provisioning algorithm individually.

D. UNDERLOADING HOST DETECTION ALGORITHM

After detecting the overloaded hosts, selecting the VMs, and sending them to the other hosts, then in this step, the underloaded hosts are determined. Because the proposed consolidation mechanism is a dynamic mechanism for different types of workload, an adaptive method is needed to determine the lower threshold and to detect underloaded hosts. For this purpose, the MRUHD algorithm is proposed.

1) MULTIPLE RESOURCES UNDERLOADING HOST DETECTION ALGORITHM (MRUHD)

MRUHD determines the lower quartile (Q_1) of the previous host utilizations as the lower threshold. Q_1 is the median of the lower half of the data set. So, the lower threshold is defined as $T_{low} = u_{(\frac{1}{4}(n+1))}$, where u is the host utilization and n is the number of data values in the data set.

Once the CPU, RAM, and BW utilizations are lower than T_{low}, the host is underloaded. Based on the vector magnitude squared (14), as shown at the bottom of the next page, which is used for calculating the magnitude of different dimensions, (15), as shown at the bottom of the next page, is provided to sort the underloaded hosts in increasing order.

In this step, the system has a list of underloaded hosts, but all the hosts will not be consolidated. First, the system needs to check that all the VMs of each host can be migrated to the other hosts. Therefore, the system investigates the possibility of consolidating all the VMs to other active hosts before starting live migrations. To accept any VM, a host must meet three conditions: (1) It must not be under pressure, (2) it must have enough resources for the VM, (3) it must not be overloaded after admitting the VM. If other active hosts admit all the VMs, the host is switched to the sleep mode, and its VMs are included in the migration list; otherwise, the host remains active. This process is iteratively repeated for all underloaded hosts. Algorithm 7 shows the MRUHD algorithm.

Algorithm 5 VM Placement (First Phase)

```
Input:  hostList, selectedVMlist
Output: allocation of VMs
(1)  selectedVMlist.sortDecreasing();// based on the
     amount of RAM;
(2)  foreach h in hostList do
(3)    if (lowrthreshold<currentUtil<pre-threshold)
       then
(4)      normalHostList←host;
(5)    else if (currentUtil < lowrthreshold)
(6)      underloadedhostList←host;
(7)    end if
(8)  end for
(9)  foreach VM in selectedVMlist do
(10)   minMark←MAX;
(11)   selectedHost←null;
(12)   foreach h in normalHost do
(13)     estimate utilAfterPlacement;
(14)     if (utilAfterPlacement < upperthreshold) then
(15)       estimate Mark by Eq. (13);
(16)       if (Mark < minMark) then
(17)         selectedHost←host;
(18)         selectedHostlist←selectedHost;
(19)         minMark←Mark;
(20)       end if
(21)     end if
(22)   end for
(23)   if (selectedHost=null) then
(24)     foreach h in underloadedHostList do
(25)       line 13 to 21;
(26)     end for
(27)   end if
(28)   if (selectedHost=null) then
(29)     selectedHostlist←new host;
(30)   end if
(31) end for
(32) return selectedhostlist;
```

Algorithm 6 VM Placement (Second Phase)

```
Input:  hostList, selectedVMlist
Output: allocation of VMs
(1)  selectedVMlist.sortDecreasing();// based on the
     amount of RAM;
(2)  foreach h in hostList do
(3)    if (currentUtil < pre-threshold) then
(4)      normalHostList←host;
(5)    else if (currentUtil < lowrthreshold)
(6)      underloadedhosList←host;
(7)      if (host admitted any VM in the first phase)
         then
(8)        receivedVMlist←host;
(9)      else
(10)       otherHostslist←host;
(11)     end if
(12)   end if
(13) end for
(14) foreach VM in selectedVMlist do
(15)   minMark←MAX;
(16)   selectedHost←null;
(17)   foreach host in normalHost do
(18)     estimate Utilafterplacement;
(19)     if (Utilafterplacement < upperthreshold) then
(20)       estimate Mark by Eq. (13);
(21)       if (Mark < minMark) then
(22)         selectedHost←host;
(23)         selectedHostlist←selectedHost;
(24)         minMark←Mark;
(25)       end if
(26)     end if
(27)   end for
(28)   if (selectedHost=null) then
(29)     foreach h in receivedVMlist do
(30)       line 18 to 26;
(31)     end for
(32)   end if
(33)   if (selectedHost=null) then
(34)     foreach host in otherHostslist do
(35)       line 18 to 26;
(36)     end for
(37)   end if
(38)   if (selectedHost=null) then
(39)     selectedHostlist←new host;
(40)   end if
(41) end for
(42) return selectedhostlist;
```

E. THE PROPOSED DYNAMIC VM CONSOLIDATION MECHANISM (PCM)

The proposed VM consolidation mechanism is comprised of four algorithms: overloading host detection algorithm, VM selection algorithm, VM placement algorithm, and underloading host detection algorithm. PCM is a combination of the best-presented algorithms for each of the four sub-mechanism introduced in the previous sections. PCM is dynamic because it uses dynamic thresholds instead of fixed-value thresholds, which makes it implicational for real, unpredictable workloads common in cloud environments. Also, the mechanism is adaptive because it automatically adjusts its behavior based on the analyses of historical data of

$$Z = \sqrt{a^2 + b^2 + c^2} \tag{14}$$

$$Z = \sqrt{(Utilization(CPU))^2 + (Utilization(RAM))^2 + (Utilization(BW))^2} \tag{15}$$

Algorithm 7 Under Loading Host Detection Algorithm

Input: *hostList, hostVMlist*
Output: *VMmigrationList*

```
(1)  foreach h in hostList do
(2)    if (h.util_CPU)<T_low(CPU) && (h.util_RAM) < T_low(RAM) && (h.util_BW) < T_low(BW) then
(3)      underloadingList←h; // host is under loaded
(4)  end for
(5)  foreach h in underloadingList do
(6)    utilC←(allocatedMips/TotalMips)^2;
(7)    utilR←(allocatedRam/TotalRam)^2;
(8)    utilB←(allocatedBw/TotalBw)^2;
(9)    Util←√(utilC + utilR + utilB);
(10)   underloadingList.sortIncreasingUtil();
(11) end for
(12) foreach h in underloadingList do
(13)   foreach VM in hostVMlist() do
(14)     foreach host in hostList do
(15)       if(host∉underPressureList) then
(16)         if ((host has enough CPU, RAM and BW) && (Not overloaded after VM migration)) then
(17)           VMmigrationList←h.VM;
(18)           hVMlist←hVMlist - h.VM;
(19)           break;
(20)         end if
(21)       end if
(22)     end for
(23)   end for
(24)   if(hVMlist=null) // after checking all VMs for each host;
(25)     return VMmigrationList;
(26)   end if
(27) end for
```

resource utilization for any application with different workload patterns. Finally, the proposed mechanism is online because the algorithms are performed run time and make an action in response to each request.

V. PERFORMANCE EVALUATION

In this section, we present the simulation results of our proposed VM consolidation mechanism. The proposed mechanism is presented for general cloud environments such as IaaS. Therefore, it should be evaluated on a large-scale, virtualized data center infrastructure. Since it is supposed to assess the VM consolidation mechanism as a repeatable experiment, conducting the experiment on a real cloud environment is difficult. Hence, simulation is a desirable choice for evaluating the proposed mechanism. The CloudSim toolkit [23] is chosen as the simulation platform because it supports energy-efficient strategies in cloud resource provisioning and also supports the ability to simulate applications that have dynamic workloads.

A. EXPERIMENTAL SETTING

To evaluate the proposed VM consolidation mechanism, we have simulated a data center comprising 800 physical servers. These servers are heterogeneous and have two server configurations: half consist of HP ProLiant ML110 G4 (Intel Xeon 3040, dual-core 1860 MHz, 4 GB, 1 Gbps), and the rest are HP ProLiant ML110 G5 (Intel Xeon 3075, dual-core 2660, 4 GB, 1 Gbps). As the target of this study is evaluating the effect of the proposed VM consolidation mechanism, servers with less resource capacity are more beneficial because these servers can be overloaded faster by lighter workloads. VMs can be run on any core, and they are not tied to any specific core. Four types of VMs corresponding to Amazon EC2 instance types are used as Micro instance (500 MIPS, 613 MB), Small Instance (1000 MIPS, 1.7 GB), Extra large Instance (2500 MIPS, 3.75 GB), High-CPU Medium Instance (2500 MIPS, 0.85 GB). Using NAS, live VM migration is enabled in the system, with no need to use direct-attached storage. This kind of storage decreases migration overhead because there is no need to copy the disk content.

Since simulation can be more applicable using data coming from real systems, real workload traces collected from the CoMon system are employed in the simulation. The CoMon project [24] creates a monitoring system for PlanetLab [25] and aims to provide information about monitoring statistics for both users and administrators. Every 5 minutes, CoMon collects workload data on roughly 400-450 active PlanetLab nodes, and 200-250 active experiments running on PlanetLab. The workload encompasses resource utilization by more than 1000 VMs from more than 500 physical hosts located around the world. Data are gathered from 10 days which are chosen randomly between March and April 2011, as shown in Table 1.

TABLE 1. Selected trace-based workloads.

Workload	No. of VMs	Workload	No. of VMs
20110303	1052	20110403	1463
20110306	898	20110309	1358
20110309	1061	20110411	1233
20110322	1516	20110412	1054
20110325	1078	20110420	1033

Each VM's workload trace is randomly dedicated to a VM during the simulation. The physical servers measure the resource usage by the VMs on a 5-minute interval. In other words, the proposed mechanism must be executed every 5 minutes based on the information provided by workload traces. The experiment runs 10 times for each algorithm and the median value is calculated and showed in terms of each of the performance metrics.

B. PERFORMANCE METRICS

Several performance metrics are used to evaluate the efficiency of the proposed VM consolidation mechanism, as follows: the number of VM migration, Performance Degradation

TABLE 2. Collected data on power consumption from the SPECpower benchmark [8].

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

due to Migration (PDM), SLA violation Time per Active Host (SLATAH), SLA Violation (SLAV), energy consumption and Energy and SLA Violation (ESV). PDM measures the degradation of system performance caused by VM migrations and can be calculated as follows:

$$PDM = \frac{1}{N}\sum_{i=1}^{N}\frac{(P_r - P_a)}{P_r} \tag{16}$$

Where N is number of VMs; P_r is the performance requested by VMs from the available hosts, and P_a is the performance allocated to the VMs. SLATAH is defined as the percentage of time active hosts experienced 100% utilization. It is notable that SLA is satisfied when the total performance demanded by the applications inside a VM is accomplished. In the case a host capacity is being 100% utilized, the host might not fully serve the VMs at the demanded performance level, and this leads to an SLA violation. SLATAH can be calculated as follows:

$$SLATAH = \frac{1}{N}\sum_{i=1}^{N}\frac{Tf_i}{Ta_i} \tag{17}$$

Where N is number of hosts; Tf_i indicates the total time during which hosts have been fully utilized and Ta_i is the total time in which hosts have been in active mode.

The combination of two previous metrics provides the main SLA violation metric, defined as follows:

$$SLAV = SLATAH \times PDM \tag{18}$$

This metric measures both performance degradation due to VM migrations and due to host overloading. Based on previous research [26], most of the energy consumption by servers is determined by the CPU, RAM, power supplies, disk storage, and cooling systems. Moreover, some studies [27], [28] proved that power consumption of physical hosts can be precisely defined by a linear relationship between CPU usage and power consumption. In this study, the real data gathered from the SPECpower benchmark [29] are used as data on power consumption for two types of hosts employed in the simulation. Based on the collected data, each type of server consumes a certain amount of energy (in Watts) in terms of its CPU utilization. Table 2 shows the collected data on power consumption used during the simulation. ESV - the last metric - evaluates the proposed VM consolidation mechanism based on both energy consumption and SLA violation rate. ESV is calculated as follows:

$$ESV = Energy\ Consumption \times SLAV \tag{19}$$

C. SIMULATION RESULTS

PCM should consist of the best algorithms proposed in the previous sections. Hence, first, we compared three VM selection policies (MPR, TPT, and V-VMs) to find the most efficient policy. Fig. 5 shows the amount of energy consumed by the data center, and Fig. 6 shows SLAV for the proposed VM selection policies. The results reveal that V-VMs obtains the minimum energy consumption and SLA violation compared with other policies. V-VMs improves energy use by 3% and 15% compared with MPR and TPT, respectively. Also, V-VMS enhances SLAV by 46% and 28% compared with MPR and TPT, respectively. These improvements illustrate that the V-VMs algorithm efficiently manages host utilization so that the hosts gain the most capacity of their utilizations while guaranteeing SLAs are met, which is done by selecting the adequate VMs to be migrated.

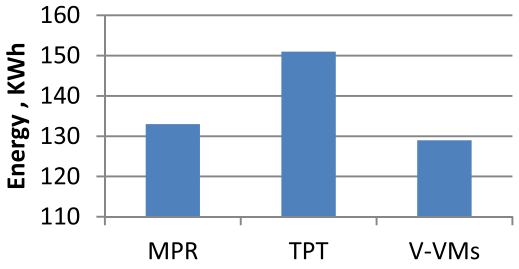

FIGURE 5. Power consumption comparison of the proposed VM selection policies.

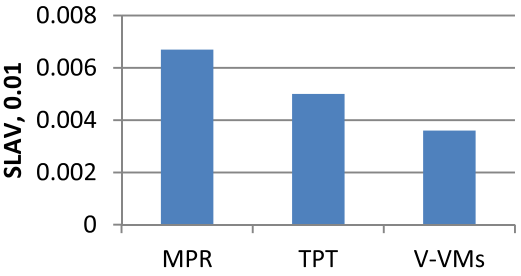

FIGURE 6. SLA violation comparison of the proposed VM selection policies.

Based on the results above, PCM is comprised of four algorithms: IWLR, V-VMS, BRB, and MRUHD. To evaluate the efficiency of the proposed mechanism, we compare it with three benchmark mechanisms: LR-RS, LR-MC, and LR-MMT [8]. The number of VM migrations incurred by PCM compared to the benchmark algorithms is shown in Fig. 7(a). The results indicate that PCM sent only

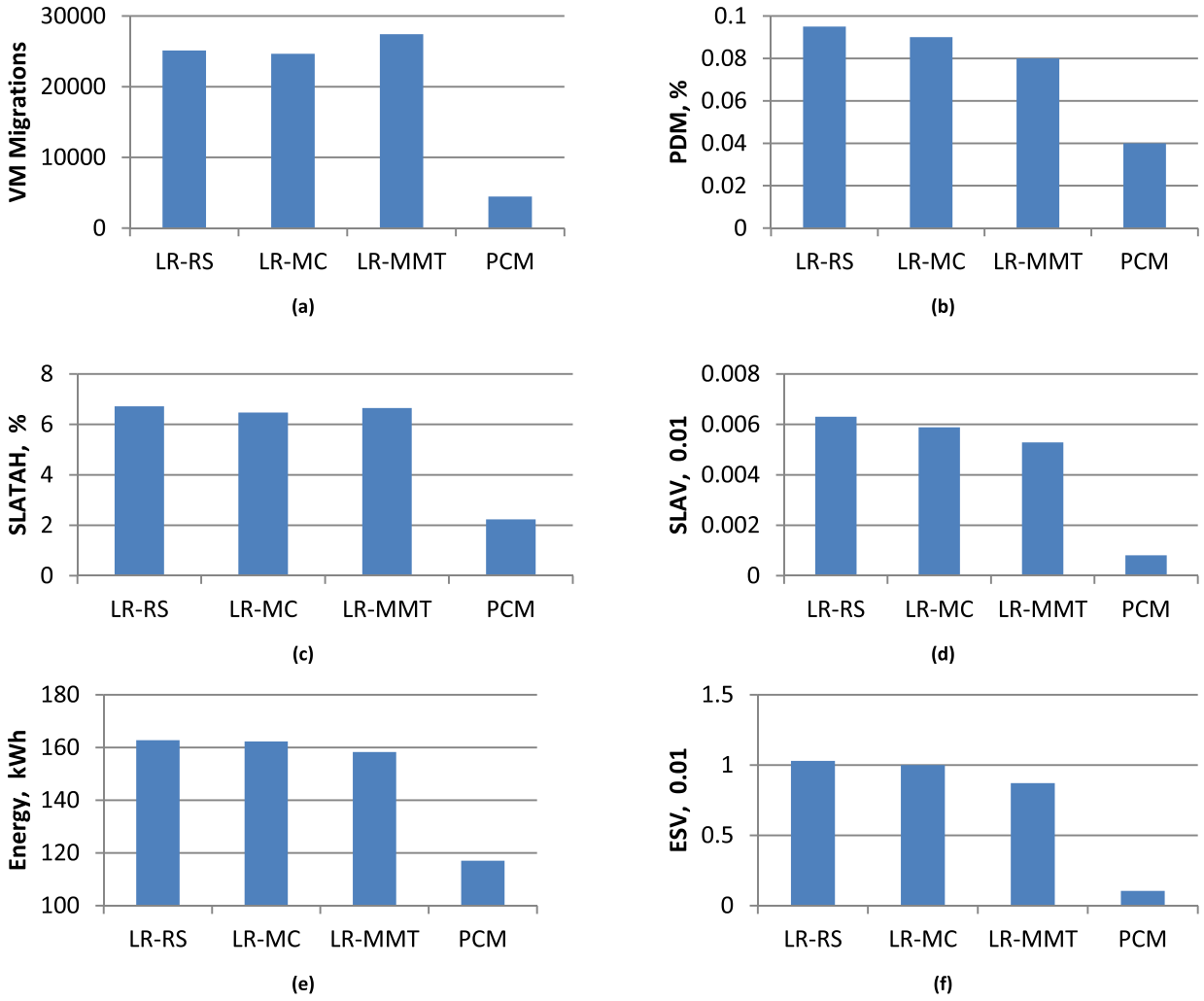

FIGURE 7. Simulation results obtained by the proposed mechanism vs. benchmark mechanisms. (a) Number of VM Migrations using PCM vs. benchmark mechanisms. (b) PDM using PCM vs. benchmark mechanisms. (c) SLATAH using PCM vs. benchmark Mechanisms. (d) SLAV using PCM vs. benchmark mechanisms. (e) Energy consumption using PCM vs. benchmark algorithms. (f) ESV using PCM vs. benchmark mechanisms.

4462 VMs to other hosts during the simulation, so compared with other benchmark mechanisms, PCM dramatically decreases the number of VM migrations. Live VM migration imposes an overhead on the system; cloud administrators set a limiting number of migrations depending on the acceptable VM migration overhead. Hence, a mechanism which requires fewer migrations to consolidate the VMs is preferred. PCM outperforms LR-RS by 82%, LR-MC by 81%, and LR-MMT by 84% in terms of the number of VM migrations.

Fig. 7(b) shows the experimental results in terms of PDM. It can be seen that PCM significantly reduces PDM compared to the other mechanisms because PCM takes into account the benefits of the V-VMs and BRB algorithms. V-VMs selects only the necessary VMs which are encountered with SLA violation and are urgent to be migrated. This issue can avoid more violation rate for those VMs and can reduce unnecessary migrations as well. Also, BRB as the VM placement algorithm helps PCM select the destination hosts regarding not only CPU but also RAM and BW, by which a more precise placement is performed, avoiding the replacement of VMs because of unsuccessful migrations. PCM improves LR-RS by 58%, LR-MC by 56%, and LR-MMT by 50% in terms of PDM. Fig. 7(c) shows that PCM outperforms SLATAH rather than benchmark algorithms. This improvement can be explained by the fact that PCM is comprised of the IWLR algorithm, which can reduce SLATAH because it aims to detect overloaded hosts before a violation occurs. It determines two thresholds to make sure that increasing the host utilization does not lead to a future violation. Therefore, the amount of time hosts are at their full capacity is reduced. On the other hand, the BRB algorithm first checks the available capacity of the hosts and possibility for VM migration before placement, which helps avoid violations due to host overloading after receiving the migrated VMs. PCM outperforms LR-RS by 67%, LR-MC by 65%, and LR-MMT by 66 % in terms of SLATAH.

TABLE 3. Simulation results of the proposed mechanism and the benchmark mechanisms (median values).

Algorithms	Number of VM Migration	PDM	SLATAH	SLAV ($\times 10^{-2}$)	Energy saving	ESV ($\times 10^{-2}$)
LR-RS	25118	0.096	6.72	0.0066	160.05	1.03
LR-MC	24666	0.095	6.47	0.0061	161.33	1
LR-MMT	27418	0.08	6.65	0.0053	161.93	0.87
PCM	4462	0.039	2.235	0.001	117.33	0.105

TABLE 4. Summary of the improvement percentages for the proposed mechanism compared to the benchmark mechanisms.

Algorithms	Improvement in the number of VM Migration (%)	Improvement in PDM (%)	Improvement in SLATAH (%)	Improvement in SLAV (%)	Improvement in energy saving (%)	Improvement in ESV (%)
LR-RS	82	58	67	87	28	90
LR-MC	81	56	65	86	28	89
LR-MMT	84	50	66	85	26	88

As described in the previous performance metrics, PCM improves other mechanisms in terms of PDM and SLATAH. Since SLAV is a combination of PDM and SLATAH, it is expected that the proposed mechanism decreases SLAV as well. Fig. 7(d) demonstrates that PCM dramatically reduces SLA violations compared to other mechanisms. PCM improves LR-RS by 87%, LR-MC by 86%, and LR-MMT by 85 % in terms of SLAV.

As the next performance metric, energy consumption is evaluated in Fig. 7(e). PCM consumes only 117 kWh during the experiment, so it outperforms the benchmark mechanisms up to 28%. The reason is that MRUHD algorithm can switch more underloaded hosts to the sleep mode, which lowers power consumption compared to the idle state, leading to more energy savings by physical nodes.

Finally, the results of ESV achieved by PMC and the benchmark mechanisms are shown in Fig. 7(f). PCM outperforms the other mechanisms in terms of both SLAV and energy consumption, up to 87% and 28%, respectively. Therefore, it is expected that PCM improves ESV compared to other mechanisms. The results show that PCM outperforms LR-RS by 90%, LR-MC by 89%, and LR-MMT by 88% in terms of ESV. The simulation results are summarized in Table 3.

The summary of the evaluation results for comparing the PCM mechanism to the benchmark mechanism is shown in Table 4. Based on the results, PCM significantly outperforms all benchmark algorithms in all six performance metrics. The results obtained by PCM in terms of SLA violations reveal a significant improvement of more than 80%. On the other hand, PCM saves energy, up to 28% more, compared to the benchmark mechanisms. These results demonstrate the proposed mechanism is quite successful in reducing energy consumption while keeping SLA violations low, which is the main target of this research. This improvement can be described by this fact that the proposed mechanism takes advantages of each algorithm. Furthermore, PCM consolidates VMs based on three resources, CPU, RAM, and BW, which provides more efficient algorithms against benchmark mechanisms.

VI. CONCLUSION AND FUTURE WORK

Dynamic consolidation of virtual machines using live migration and switching idle servers to the sleep mode allows cloud providers to optimize resource utilization and reduce energy consumption. However, aggressive VM consolidation can lead to performance degradation. Several energy-efficient techniques have already been presented in the literature, but the rate of SLA violations is still significantly high. Furthermore, the current algorithms consider CPU as the only factor in VM consolidation. In this study, after investigating the previous studies, we proposed an energy-efficient and SLA-aware VM consolidation mechanism; its goal is reducing the energy consumption of a data center while trying to guarantee required system performance under SLA constraints regarding CPU, RAM, and BW. To evaluate the proposed VM consolidation mechanism, CloudSim was chosen as the simulation platform. An extensive simulation was carried out on a large-scale experiment setup using workloads traced from more than 1000 PlanetLab VMs. The experiment results have shown the effectiveness of the proposed mechanism compared to the benchmark algorithms. PCM outperformed all the benchmark algorithms in terms of energy consumption and SLA up to 28% and 87%, respectively.

Another important factor in consolidation mechanism which can be considered for the future work is the network topology. VMs running the application may need to communicate with each other due to the workload dependencies. Therefore, monitoring the VM's communications and then more efficient allocation of VMs to adequate servers can decrease the overhead of data transfer and energy consumption by the network devices. On the other hand, cloud environments consist of heterogeneous servers with different characteristics and capacities which provide various levels of performance. Therefore, a performance-aware strategy which can deal with various workloads provided by the applications running on the system can improve energy-efficient VM consolidation mechanism in cloud data centers. This was not a part of our research scope but can be considered in future works.

APPENDIX

TABLE 5. Comparison of the related works.

Research	Resources	Objectives	Energy-efficient technique
Verma et al. [6]	CPU	Reduce power consumption , Satisfy QoS	VM consolidation , DVFS, Power switching
Beloglazov et al. [7]	CPU	Reduce power consumption , Satisfy SLA	VM consolidation
Beloglazov and Buyya [8]	CPU	Reduce power consumption , Satisfy SLA	VM consolidation
Mhedheb et al. [9]	CPU	Reducing power consumption , Load and temperature balancing	DVFS , VM consolidation
Taheri and Zamanifar [10]	CPU	Reduce power consumption , Reduce incomplete migrations	VM consolidation
Shi et al. [11]	CPU - memory	Reduce power consumption	VM consolidation
Keller et al. [12]	CPU	Reduce power consumption , Reduce dropped request	VM consolidation
Farahnakian et al. [13]	CPU	Reduce power consumption , Satisfy SLA	VM consolidation
Takouna et al. [14]	CPU	Reduce power consumption , Optimize acceptance ratio job	DVFS , VM consolidation
Chawraut and Woraphon [15]	CPU	Reduce power consumption , Satisfy QoS	DVFS , VM consolidation
Lim et al. [16]	CPU	Reduce power consumption , Satisfy SLA	VM consolidation
Wang et al. [17]	CPU	Reduce power consumption , Satisfy SLA	VM consolidation
Zhou et al. [18]	CPU	Reduce power consumption , Satisfy SLA	VM consolidation

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, and R. Katz, "Above the clouds: A Berkeley view of cloud computing," UC Berkeley Reliable Adaptive Distrib. Syst. Lab., Berkeley, CA, USA, Tech. Rep. UCB/EECS-2009-28, 2009.

[2] B. P. Rimal, A. Jukan, D. Katsaros, and Y. Goeleven, "Architectural requirements for cloud computing systems: An enterprise cloud approach," *J. Grid Comput.*, vol. 9, no. 1, pp. 3–26, 2011.

[3] J. Koomey, *Growth in Data Center Electricity Use 2005 to 2010*. vol. 9. El Dorado Hills, CA, USA: Analytical, 2011.

[4] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *Proc. Conf. Power Aware Comput. Syst.*, vol. 10. 2008, pp. 1–5.

[5] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges," in *Proc. Int. Conf. Parallel Distrib. Process. Techn. Appl. (PDPTA)*, Jul. 2010, pp. 1–12.

[6] A. Verma, P. Ahuja, and A. Neogi, "pMapper: Power and migration cost aware application placement in virtualized systems," in *Proc. ACM/IFIP/USENIX Int. Conf. Distrib. Syst. Platforms Open Distrib. Process.*, Dec. 2008, pp. 243–264.

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

[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. Exper.*, vol. 24, no. 13, pp. 1397–1420, 2012.

[9] Y. Mhedheb, F. Jrad, J. Tao, J. Zhao, J. Kolodziej, and A. Streit, "Load and thermal-aware VM scheduling on the cloud," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.*, Dec. 2013, pp. 101–114.

[10] M. M. Taheri and K. Zamanifar, "2-phase optimization method for energy aware scheduling of virtual machines in cloud data centers," in *Proc. IEEE Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2011, pp. 525–530.

[11] L. Shi, J. Furlong, and R. Wang, "Empirical evaluation of vector bin packing algorithms for energy efficient data centers," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2013, pp. 9–15.
[12] G. Keller, M. Tighe, H. Lutfiyya, and M. Bauer, "An analysis of first fit heuristics for the virtual machine relocation problem," in *Proc. 8th Int. Conf. Workshop Syst. Virtualization Manage. (SVM) Netw. Service Manage. (CNSM)*, Oct. 2012, pp. 406–413.
[13] F. Farahnakian, P. Liljeberg, and J. Plosila, "Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning," in *Proc. 22nd Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process. (PDP)*, Feb. 2014, pp. 500–507.
[14] I. Takouna, W. Dawoud, and C. Meinel, "Energy efficient scheduling of HPC-jobs on virtualize clusters using host and VM dynamic configuration" *ACM SIGOPS Oper. Syst. Rev.*, vol. 46, no. 2, pp. 19–27, 2012.
[15] W. Chawarut and L. Woraphon, "Energy-aware and real-time service management in cloud computing," in *Proc. 10th Int. Conf. Elect. Eng./Electron., Comput., Telecommun. Inf. Technol. (ECTI-CON)*, May 2013, pp. 1–5.
[16] M. Y. Lim, F. Rawson, T. Bletsch, and V. W. Freeh, "PADD: Power aware domain distribution," in *Proc. 29th IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2009, pp. 239–247.
[17] X. Wang, X. Liu, L. Fan, and X. Jia, "A decentralized virtual machine migration approach of data centers for cloud computing," *Math. Problems Eng.*, vol. 2013, Jul. 2013, Art. no. 878542.
[18] Z. Zhou, Z. Hu, and K. Li, "Virtual machine placement algorithm for both energy-awareness and SLA violation reduction in cloud data centers," *Sci. Program.*, vol. 2016, Mar. 2016, Art. no. 5612039.
[19] J. B. Guerard, Jr., *Introduction to Financial Forecasting in Investment Analysis*. New York, NY, USA: Springer, 2013.
[20] S. Weisberg, *Applied Linear Regression*, vol. 528. Hoboken, NJ, USA: Wiley, 2005.
[21] W. S. Cleveland, "Robust locally weighted regression and smoothing scatterplots," *J. Amer. Statist. Assoc.*, vol. 74, no. 368, pp. 829–836, 1979.
[22] W. S. Cleveland, and C. Loader, "Smoothing by local regression: Principles and methods," in *Statistical Theory and Computational Aspects of Smoothing*, Heidelberg, Germany: Physica-Verlag HD, 1996, pp. 10–49.
[23] R. Buyya and M. Murshed, "GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency Comput., Pract. Exper.*, vol. 14, nos. 13–15, pp. 1175–1220, 2002.
[24] 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.
[25] B. Chun *et al.*, "PlanetLab: An overlay testbed for broad-coverage services," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 3–12, Jul. 2003.
[26] L. Minas and B. Ellison, *Energy Efficiency for Information Technology: How to Reduce Power Consumption in Servers and Data Centers*. Santa Clara, CA, USA: Intel Press, 2009.
[27] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," *Cluster Comput.*, vol. 12, no. 1, pp. 1–15, Mar. 2009.
[28] X. Fan, W. D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proc. 34th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2007, pp. 13–23.
[29] *The SPECpower Benchmark*, accessed on Jan. 2016. [Online]. Available: http://www.spec.org/power_ssj2008/

MOHAMMAD ALI KHOSHKHOLGHI received the master's degree in computer science, in Iran, in 2010. He is currently pursuing the Ph.D. degree with the Faculty of Computer Science and Information Technology, University Putra Malaysia. His current research interests include cloud computing, green computing, wireless sensor networks, network security and computer networks. He has authored several journal papers.

MOHD NOOR DERAHMAN is currently a Lecturer with the Department Communication Technology and Network, Faculty of Computer Science and Information Technology, University Putra Malaysia. His research interests include Computer Networks and network security.

AZIZOL ABDULLAH received the M.Sc. degree in engineering (telematics) from The University of Sheffield, U.K., in 1996 and the Ph.D. degree in distributed system from University Putra Malaysia, Malaysia, in 2010. He is currently a Senior Lecturer with the Department Communication Technology and Network, Faculty of Computer Science and Information Technology, University Putra Malaysia. He has been appointed as a Fellow Researcher with ITU-UUM Asia Pacific Centre of Excellence for Rural ICT Development (ITU-UUM). His main research areas include cloud and grid computing, network security, wireless and mobile computing and computer networks.

SHAMALA SUBRAMANIAM received the B.S. degree in computer science from University Putra Malaysia in 1996, and the M.S. and Ph.D. degrees from University Putra Malaysia in 1999 and 2002, respectively. She is currently a Professor with the Department Communication Technology and Network, Faculty of Computer Science and Information Technology, University Putra Malaysia. Her research interests include Computer Networks, Simulation and Modelling, and Scheduling and Real-Time Systems.

MOHAMED OTHMAN was a Deputy Director of the Information Development and Communication Center, who involved in monitoring the UMPNet network campus, uSport Wireless Communication project and UPM Server farm. He is currently a Professor in computer science with the Department of Communication Tech and Network, Faculty of Computer Science and Information Technology, University Putra Malaysia. His main research interests are in the fields of parallel and distributed algorithms, high-speed networking, network design and management (network security, wireless and traffic monitoring), and scientific computing.