Received: 23 January 2019 | Revised: 18 August 2019 | Accepted: 10 September 2019

DOI: 10.1002/spe.2764

RESEARCH ARTICLE

WILEY

Adaptive Markov-based approach for dynamic virtual machine consolidation in cloud data centers with quality-of-service constraints

Hossein Monshizadeh Naeen | **Esmaeil Zeinali** | **Abolfazl Toroghi Haghighat**

Faculty of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran

Correspondence
Esmaeil Zeinali, Faculty of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin 34185 1416, Iran.
Email: zeinali@qiau.ac.ir

Summary

Dynamic virtual machine (VM) consolidation is one of the emerging technologies that has been considered for low-cost computing in cloud data centers. Quality-of-service (QoS) assurance is one of the challenging issues in the VM consolidation problem since it is directly affected by the increase of resource utilization due to the consolidations. In this paper, we take advantage of Markov chain models to propose a novel approach for VM consolidation that can be used to explicitly set a desired level of QoS constraint in a data center to ensure the QoS goals while improving system utilization. For this purpose, an energy-efficient and QoS-aware best fit decreasing algorithm for VM placement is proposed, which considers QoS objective when determining the location of a migrating VM. This algorithm employs an online transition matrix estimator method to deal with the nonstationary nature of real workload data. We also propose new policies for detecting overloaded and underloaded hosts. The performance of our proposed algorithms is evaluated through simulations. The results show that the proposed VM consolidation algorithms in this paper outperforms the benchmark algorithms in terms of energy consumption, service-level agreement violations, and other cost factors.

KEYWORDS

cloud computing, dynamic consolidation, energy efficiency, green information technology, Markov chains, virtualization

1 | INTRODUCTION

Cloud computing is a growing technology trend in the past few years, which is used to run business and consumer-based IT applications on them.[1,2] A cloud data center provides a set of resources that can be used in a shared way to run the applications.[3] Usually, most running applications operate only on a small part of the total requested resources; therefore, elasticity is one of the main required features in cloud computing environments to deliver services based on the changes in the customer demands. A major concern in flexible environments is quality-of-service (QoS) fulfillment.[4,5] On the other hand, the rapid growth in the use of cloud computing has increased energy consumption and other costs for cloud data centers.[6]

Cloud providers can employ server consolidation algorithms to manage energy consumption and service-level agreement (SLA) violations efficiently. Dynamic server consolidation is an approach that uses a combination of advanced technologies, such as virtualization and live migration, to consolidate virtual machines (VMs) on a limited subset of all available physical machines (PMs). Therefore, the total energy consumption reduces, since the number of active servers reduces and idle ones are switched off.[7] It can be concluded that green cloud computing is envisioned to achieve the efficient processing and utilization of computing infrastructure and to minimize energy consumption.[8] Minimizing the number of active PMs may lead to some QoS degradation. The QoS constraints can be defined in terms of a variety of metrics, which are formalized in the SLAs. In this work, the QoS requirements are specified as a workload-independent metric, which is discussed in a study performed by Beloglazov and Buyya[9]; therefore, the problem transforms into the minimization of energy consumption under QoS constraints. Because of the complexity of the problem, it is usually divided into smaller subproblems:

- *Overloaded hosts detection:* Detecting the hosts that are violating SLA agreements or are prone to SLA violations.
- *Underloaded hosts detection:* Detecting the hosts that all of their residing VMs can be migrated out and thus they can get switched off.
- *VM selection:* Determining the VMs that should be migrated from overloaded hosts.
- *VM placement:* Determining new destinations for VMs that are selected for migration.

Whenever a host becomes overloaded or underloaded, some VMs should be relocated. The VM placement algorithm determines the new placement of a VM. Several researches have introduced effective VM placement algorithms that may use one of: heuristic methods (eg, TOPSIS power and SLA–aware allocation (TPSA),[1] stochastic process–based best fit decreasing (SBBFD),[10] and greedy approach[11]), metaheuristic methods (eg, simulated annealing,[2] ant colony optimization,[12] and genetic algorithm[13]), or other methods (eg, mathematical programming approach[14] and constraint programming approach[15]). The deficiency of these methods, which leads to suboptimal placements, is that they do not allow to explicitly set a QoS objective in VM placement algorithm. In contrast, in this paper, a novel VM placement algorithm is proposed, which enables a system administrator to explicitly set a QoS objective when it performs a new VM placement.

The present study provides a workload adaptive Markov-based dynamic server consolidation policy to minimize energy consumption and satisfy a specified level of QoS. It is focused on the theoretical study of VM consolidation problem in this work, and simulations are used to evaluate the proposed algorithms. Note that consolidation algorithms can be practically implemented on existing open-source cloud management platforms, such as OpenStack*, Eucalyptus†, and snooze‡. In this regard, some studies have worked on practical implementations.[16,17] This study is mainly inspired by a work of Beloglazov and Buyya[9] on a cloud management system, in which they use Markov decision processes to handle overloaded hosts; accordingly, workloads on PMs are considered as Markov processes. It is necessary to note that this kind of modeling requires assuming that workloads on PMs satisfy the Markov property, which implies memoryless state transitions. This assumption must be taken into account in an assessment of the applicability of the proposed algorithms in this paper to a particular system. In this regard, the PlanetLab workload traces[18] are chosen for our experiments since Beloglazov and Buyya[9] have considered them as workloads with Markov property. However, the assumption of memoryless state transitions may not be accurate in many systems, and it is possible to envision systems, in which future states depend on more than one past state. It is focused on the placement algorithm in this paper, also detecting overloaded and underloaded host subproblems that are reinspected, and new algorithms are suggested for them. The main contributions in this paper are as follows:

1. Proposing an online algorithm named online transition matrix estimator (OTEST) that deals with nonstationary Markov processes, detects the workload changes in an online manner, and estimates the one-step transition matrix based on the current stationary workload on a host.
2. Analyzing the required conditions for the placement of VMs on hosts according to the QoS objective.
3. Introducing a new concept called qualified PMs (QPMs), which can be used in an environment with nonstationary workloads to determine the list of suitable hosts (ie, hosts that will satisfy the QoS objective) for a given VM.
4. Introducing three overload detection policies to ensure QoS establishment on physical hosts and proposing a novel underload detection policy to reduce energy consumption.

The experiment results indicate that our approach reduces energy consumption up to 15% when compared to the best of benchmark algorithms in this metric. The results also indicate that the proposed algorithms in this paper lead to fewer

*The OpenStack Cloud platform. http://openstack.org/
†Eucalyptus Cloud software. https://www.eucalyptus.cloud/
‡The Snooze Cloud manager. http://snooze.inria.fr/

SLA violations; thus, the simultaneous optimization of energy consumption and SLA violation metric (ECSV) decreases noticeably. Furthermore, the total cost in terms of energy consumption, SLA violations, and the number of migrations metric (ECSVM) has significantly reduced according to the paired *t* tests.

The remainder of the paper is organized as follows: Related works are reviewed in Section 2. Section 3 describes the system model including the data center model, the QoS metric used in this study, and the adaptive approach used for real workload data modeling with Markov chains, which includes an OTEST method to deal with the nonstationary characteristics of the workloads. A novel QoS-aware VM placement algorithm named energy-efficient and QoS-aware BFD (EQBFD) is presented in Section 4. We discuss the QoS assurance policies followed by the low-loaded host detection policy in Sections 5 and 6, respectively. Section 7 describes performance evaluation process, which includes the experimental design and setup using CloudSim simulator, the evaluation metrics, and the analysis of results. Finally, in Section 8, a summary of the work and possible future directions of this research is presented.

2 | RELATED WORKS

As stated by Arianyan et al,[19] there is a wide area of research in the resource management field in cloud computing. Therefore, in this section, the research works that have closer relations with this study are reviewed.

Beloglazov and Buyya[20] have presented novel adaptive heuristics for the VM consolidation problem. To find overloaded hosts, they have proposed four different methods: median absolute deviation, interquartile range, local regression, and robust local regression. These methods did not consider desired levels of QoS to identify overloaded hosts. Also, they have introduced a power-aware BFD (PABFD) algorithm for VM placements, which places each VM on a host that provides the least increase in power consumption. PABFD cannot be employed to set a QoS objective when performing the placements; therefore, its employment in a cloud system may cause the physical hosts to become overloaded many times, and thus the number of VM migrations due to host overloading increases considerably.

In another work,[9] Beloglazov and Buyya have recommended an innovative solution for detecting overloaded hosts to satisfy a certain level of QoS. They have proposed an algorithm that heuristically adapts to workload variations using a multisize sliding window workload estimation technique. The main difference between their study[9] and our work is that they have focused on the overloaded host detection only, while we have focused on VM placement problem and also consider new solutions for other consolidation subproblems. Besides, our solution includes an approach that dynamically detects the longest estimation trace length, in contrast to the multisize sliding window technique (which selects the estimation trace length from one of the predefined sizes) used by Beloglazov and Buyya.[9]

In a different study,[10] Naeen et al have developed a stochastic process–based VM consolidation policy to reduce the total costs of data centers. It is assumed that the workloads on PMs are stochastic processes with normal distributions, and based on this assumption, they proposed a VM placement algorithm named SBBFD that considers the overload probabilities before VM placements. The main difference between the stochastic-based consolidation policy and the present study is that the VM placement and overload detection algorithms suggested in the stochastic-based approach cannot be used to set a level of QoS objective for the consolidation problem. Besides, they apply a sliding window method but with single-size windows for workload data modeling.

Arianyan et al[1] have proposed a multicriteria algorithm named TPSA, which considers multiple criteria in its decision-making for VM placement. In their work, three underload detection policies (available capacity (AC), migration delay (MDL), and TOPSIS AC, number of VMs, and MDL (TACND)) are suggested to determine underloaded PMs. They have also introduced a fuzzy system model in their next study,[19] in which hosts are scored according to different criteria such as available resources, power increase, number of VMs, etc. A machine with the best score is selected as the new host for a VM. Despite the reduction in SLA violations, the policies proposed in their aforementioned studies cannot be used to set a QoS objective.

Horri et al[21] have proposed a QoS-aware VM consolidation approach, which considers the utilization history of VMs to decide about the location of a VM. Their proposed algorithm is based on the idea given in a study by Verma et al[22]; the idea is that there is a high probability that a server becomes overloaded if there is a high correlation between the VMs residing on that host. According to this idea, Horri et al[21] selected a host with high utilization and minimum correlation (called UMC policy) as the destination of a migrating VM. They have also proposed a VM-based dynamic threshold (VDT) algorithm to detect underloaded hosts. The experimental results of their proposed approach show that it reduces SLA violations.

Abadi et al[23] have proposed a self-adaptive approach for VM consolidation problem based on a probabilistic model of the data center by employing queuing theory. In their research, a QoS metric is defined according to the response time of the arriving tasks. They assume each VM executes only one of the predefined types of arrival requests; therefore, the system is application specific and should be content aware. In a work by Zhang et al,[24] they proposed a VM placement algorithm named QUEUE, which is also based on queuing theory. Their proposed system deals with workloads that have burstiness pattern. They model the resource requirement of each individual VM as an ON-OFF Markov chain to represent burstiness, based on which a reservation strategy via a queuing theory approach is given for each physical host. The main difference between their work and our proposed system is that QUEUE algorithm works with the assumption that workloads have burstiness pattern, and similar to Abadi et al,[23] the workload pattern is known a priori. In addition, Zhang et al[24] have assumed that the workload patterns are stationary and do not change over time; therefore, no special overload or underload detection policies are introduced in their study.

Two energy-aware consolidation heuristics with the goal of minimizing energy consumption by maximizing resource utilization are suggested in a work done by Lee and Zomaya.[25] Their heuristics assign each task to the resource on which the energy consumption for executing the task is minimized without the performance degradation of that task. They have assumed that energy consumption has a direct dependency on CPU utilization, and their energy-aware consolidation heuristics are developed based on this assumption. In a similar work, Asyabi and Sharifi[26] proposed an energy-aware heterogeneous VM scheduling algorithm. They used a set of objective functions in terms of a cost factor and then placed a set of heterogeneous VMs on a set of servers aiming to minimize the total energy consumption in the entire data center. Also, they presented a new algorithm for selecting the best VMs for migration from overloaded hosts. Similar to the work done by Lee and Zomaya[25], their proposed algorithms cannot be used to set a certain QoS constraint.

In another study,[27] Wang and Wang have proposed a performance-controlled VM consolidation method that works in two levels. In one level, there is an application level in which a model predictive controller is responsible for tuning CPU frequency and VM resizing. In the other level, there is a power optimizer that monitors the CPU resource demand of VMs in another level called cluster level. It is responsible for finding a power-efficient VM to PM migration map that satisfies QoS requirements. This VM consolidation system has been designed for specific cloud environments, such as high-performance computing, where a deadline is determined in the agreements.

Gao et al[28] have studied the dynamic energy management by considering QoS and resource utilization in the system optimally to reduce energy usage. They have presented a BFD-based heuristic providing a high-quality mapping of VMs on PMs. There are also other heuristic-based VM allocation algorithms presented in the literature (eg, availability-based VM allocation (AVL),[29] and medium-fit power-efficient decreasing (MFPED)[30]) with the goal of reducing energy consumption while trying to minimize QoS violations. The VM placement proposed in these systems cannot be used to set an explicit QoS goal by the service provider.

Our work is different from previous works since the proposed VM allocation algorithm here allows a cloud provider to explicitly set a QoS objective. To the best of our knowledge, this is the first work that the VM allocation algorithm enables this possibility for cloud environment with dynamic nonstationary workloads. In this regard, other consolidation subproblems are reinspected to match the objective of this paper. We also propose an online Markov transition matrix estimator (OTEST) algorithm that deals with nonstationary and unknown a priori workload data. OTEST uses an adaptive approach to dynamically model the behavior of utilization on PMs as Markov chains. This algorithm automatically fits the Markov chain models based on the changes in utilization processes. There are also new suggested policies for overloaded host detection to assure QoS objective satisfaction.

3 | PROPOSED SYSTEM MODEL

This study aims at proposing an approach for dynamic VM consolidation, which reduces energy consumption under QoS constraints. Saving energy without sacrificing QoS is a challenging problem for cloud providers. The proposed system in this study includes a datacenter with heterogeneous PMs on which heterogeneous VMs can be deployed. Workloads on PMs are modeled as nonstationary Markov processes. Figure 1 illustrates a high-level view of the logical organization of the proposed system.

Each physical host has a coordinator as a module of its VM monitor, which is run locally. A coordinator is responsible for continuous monitoring resource utilization, detecting changes in utilization process, modeling utilization processes as a Markov chain, and detecting the time when the host becomes overloaded or underloaded. To model the workload data,

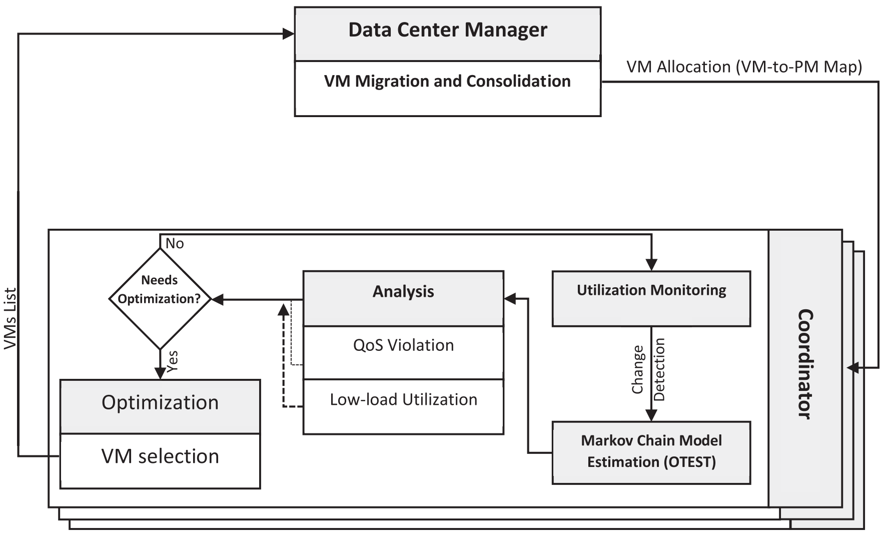

FIGURE 1 A high-level view of the logical organization of the proposed system. OTEST, online transition matrix estimator; PM, physical machine; VM, virtual machine

the coordinator employs an algorithm named OTEST to estimate the transition matrix based on the current stationary workload.

The workload on each PM may be nonstationary; thus, the transition probabilities of the Markov chain of CPU utilization on a host may change over time. The OTEST algorithm employs an online change detection method that identifies significant changes in CPU utilization process to deal with this issue; therefore, the proposed algorithm is capable of estimating the length of historical data during which the current utilization process can be assumed to be stationary. Finally, the OTEST algorithm calculates the transition matrix by applying the maximum likelihood estimation method to the current stationary workload data (more details on the OTEST algorithm are given in Section 3.2.2).

After constructing the Markov chain of utilization process, if the coordinator recognizes the host state as overloaded or underloaded, it issues an optimization request to the data center manager, in which it specifies the list of VMs that need relocation. The data center manager employs a QoS-aware VM placement algorithm named EQBFD, to specify the new location of the VMs that need optimization. To do this, the EQBFD also employs the OTEST algorithm to construct the list of QPMs for a migrating VM. Here, the OTEST is used to obtain the transition matrix of CPU utilization after a VM migration, and then the most energy-efficient QPM with the lowest AC is chosen as the best-fitting PM. Note that, in this system, the data center manager and the local coordinators take some part of the responsibility for both the energy optimization and QoS assurance:

1. Each of the coordinators is responsible for controlling the QoS objective by detecting or predicting overload conditions.
2. Each of the coordinators is responsible for controlling the energy consumption by detecting underload conditions.
3. The data center manager controls the QoS objective by allowing to set a desired QoS level in VM placement algorithm.
4. The data center manager reduces energy consumption by optimizing the number of active PMs via VM placement algorithm.

The proposed system model makes it possible to execute all the proposed algorithms that are implemented as the functions of the coordinator in distributed form; therefore, the scalability of the system improves because most of the important consolidation subproblems are performed locally and independently from other PMs.

3.1 | QoS metric

In this paper, the QoS requirements are specified as the *extended workload independent QoS metric* introduced by Beloglazov and Buyya.[9] Showing the total resource demand on a PM with d_u and the PM's capacity with C_h, whenever $C_h < d_u$, the capacity constraint of the host is said to be violated and the host is considered overloaded. Therefore, the overload time fraction (OTF) is defined as follows:

$$\mathrm{OTF}_h = \frac{\int_0^t o_h(t)dt}{\int_0^t a_h(t)dt}, \tag{1}$$

where $o_h(t)$ returns one if the host h is overloaded at time t; otherwise, it returns zero. Similarly, the function $a_h(t)$ is equal to one if the host h is active at time t. A smaller value of OTF determines a higher QoS on the host h. Using this metric, QoS can be defined as the maximum allowed value of OTF. Therefore, the QoS objective is expressed in terms of OTF:

$$\forall h \in \text{PM}s, \quad \text{OTF}_h < \rho, \tag{2}$$

where ρ is the limit on the maximum allowed OTF value. It is important to note that the performance of the applications that run on an overloaded host degrades due to the resource contention.[31,32] Various works have investigated performance interference due to resource contentions in cloud data centers. For example, Mars et al[33] have presented a methodology to accurately estimate the performance interference between colocated applications on clouds with multicore systems. Govindan et al[34] have also discussed the performance degradation due to resource contention between consolidated VMs. There exist some scheduling studies that try to guarantee the QoS goal of diverse applications in data centers, which consider the interference between collocated workloads on a server.[35-38] However, the contention issue due to the multitenancy and its effect on the QoS of individual applications that execute on VMs is out of the scope of this paper.

3.2 | Workload modeling with Markov chains

In cloud data centers, each VM that is allocated to a PM uses a part of the CPU capacity. Therefore, the set of VMs on a PM constitutes the PM's workload. In this paper, it is assumed that workload data, which are measured at discrete time steps on each PM, satisfy Markov property, and thus they have memoryless state transitions. In other words, we treat the total utilization by the set of VMs on a PM at time t as a categorical random variable; therefore, if the utilization range is categorized into a set of $|S| = k + 1$ classes, the CPU utilization can be assumed as a stochastic process from which we can construct a Markov chain with k + 1 states, ie,

$$P\left(X_{t+1} = s_{t+1} | X_t = s_t, \quad X_{t-1} = s_{t-1}, \ \dots, X_{t=0} = s_{t=0}\right) = P\left(X_{t+1} = s_{t+1} | X_t = s_t\right). \tag{3}$$

It means that the probability distribution of future state of the process depends only upon the current state. Note that each class is associated to a utilization interval, denoted by i ($s_i \in S, 0 \leq i \leq k$). For example, if we take $k = 5$, then the utilization range is divided to $s_0 = [0, 0.2), \ \dots, s_4 = [0.8, 1)$, and $s_k = s_5 = [1, \max(d_u)]$. Therefore, there is a Markov chain of utilization on each PM. Let p_{ij} be the probability of the transition from state i to state j; then, if the CPU utilization on a PM is within the range of the interval s_i at time t, the probability that it will be in the interval s_j at time $t+1$ is p_{ij}. Using the resource utilization history of a PM, the transition probability p_{ij} can be estimated by applying the maximum likelihood estimation method:

$$\hat{p}_{ij} = \frac{n_{ij}}{\sum_{j=1}^{k} n_{ij}}, \tag{4}$$

where n_{ij} is the number of transitions from state i to state j. Therefore, the one-step transition matrix $\mathbf{M}$ can be constructed as follows:

$$\mathbf{M} = \begin{pmatrix} \hat{p}_{00} & \cdots & \hat{p}_{0k} \\ & \cdots & \\ \hat{p}_{k0} & \cdots & \hat{p}_{kk} \end{pmatrix}. \tag{5}$$

3.2.1 | Adaptive workload data modeling

Equation (5) can be calculated and used for stationary Markov chains, but the problem we face in the real-world data is that they are nonstationary. Thus, the process of utilization on each host is not constant and may change over time. The solution is to approximate a nonstationary process by dividing it into a sequence of stationary processes. Such a division requires complete information on workloads and the times that their properties change. In fact, we do not have such information and decision-making can be performed only based on previous observations. Thus, heuristic methods for the approximation of such solution should be presented. To find abrupt changes in the mean level of stochastic processes, some statistical methods known as step detection are used. Since the changes may be small or the process may be corrupted by some kind of noise or other events, this makes the problem challenging and it is necessary to use appropriate methods to detect them. In this study, as the workload traces arrive over time and they are unknown a priori, an online algorithm named cumulative sum control chart (CUSUM) is employed to detect changes in Markov chains of CPU utilization on PMs.

Change detection: CUSUM is a sequential analysis technique that is used for monitoring process changes.[39] Assuming μ_0 as the utilization process mean, then the CUSUM against the sample number i is calculated as follows:

$$C_t = \sum_{j=1}^{t} (\bar{u}_j - \mu_0), \tag{6}$$

where $\bar{u}_j$ is the average of CPU utilization at the jth time interval, and C_t is called the cumulative sum up to and including the tth sample. If the utilization mean increases to some value $\mu_1 > \mu_0$, then an upward or positive shift will develop in the cumulative sum and it is called upper CUSUM shown with a statistic C_t^+. Otherwise, if the utilization mean decreases to some $\mu_1 < \mu_0$, then the value of cumulative sum will go downward and the cumulative sum is called lower CUSUM shown with another statistic C_t^-.

In this paper, we employ two-sided standardized CUSUM, which is an extension of regular CUSUM. The standardized value of $\bar{u}_t$ is calculated as follows:

$$y_t = \frac{\bar{u}_t - \hat{\mu}}{\hat{\sigma}}, \tag{7}$$

where $\hat{\mu}$ and $\hat{\sigma}$ are estimated from a training phase as the target mean and standard deviation, respectively. Therefore, the standardized CUSUMs are defined as follows:

$$C_t^+ = \max\left(0, y_t - R + C_{t-1}^+\right), \tag{8}$$

$$C_t^- = \max\left(0, y_t - R + C_{t-1}^-\right), \tag{9}$$

where the starting values are $C_0^+ = C_0^- = 0$, and R is the mean-shift detection constant for the CUSUM chart, which is known as reference value. If either C_t^+ or C_t^- exceed the decision interval H, the process is considered to be out of control, ie, a significant change in the utilization process has happened. A counter can be added to the CUSUM algorithm to indicate when the shift probably occurred. There are two advantages to standardizing the CUSUM, which we will benefit from them in our experiments. First, many CUSUMs can have the same values of R and H, and the choices of these parameters are not scale dependent (ie, they do not depend on σ). Second, a standardized CUSUM leads naturally to a CUSUM for controlling variability, ie, it can be extended to be sensitive to both mean and variance changes.[39]

Hints for CUSUM design: Generally, R is relative to the size of the shift we want to detect. Choosing the values for allowance value R and decision interval H has an important effect on detecting changes correctly and at a right time. It is usually recommended that to set these parameters values to provide good average run length performance.[39] R is often chosen about halfway between the target process mean (μ_0) and the mean that the process is considered to be out of control (μ_1). Therefore, if the shift is expressed in standard deviation (σ) units, then

$$\mu_1 = \mu_0 + \delta\sigma \Rightarrow \delta = \frac{|\mu_1 - \mu_0|}{\sigma}, \tag{10}$$

$$R = \frac{|\mu_1 - \mu_0|}{2} \Rightarrow R = \frac{\delta\sigma}{2}, \tag{11}$$

where δ is the size of the shift in standard deviation units. For instance, if the process is considered to be stationary in the interval $\mu \pm 3\sigma$ (ie, $\delta = 3$) then the allowance value is $R = \frac{3}{2}\sigma$. If either C_t^+or C_t^- exceed the decision interval, which is suggested to be about $H = 5\sigma$,[39] the Markov process has significantly changed compared to the previously known stationary process, and thus new estimations should be done (more details, reasoning and analysis are given by Montgomery[39]).

3.2.2 | Online transition matrix estimator

Figure 2 shows the high-level view of our proposed OTEST algorithm. The algorithm gets a set of CPU utilization trace as an input, and estimates the one-step transition probabilities of the current stationary utilization process as the output. The CUSUM method is used to detect abrupt changes in the utilization process. Initial observations (30 time steps in our experiments) after each change point are considered as training data to acquire the target mean and standard deviation for CUSUM. The number of samples from the last detected change point to the current time is considered estimation trace

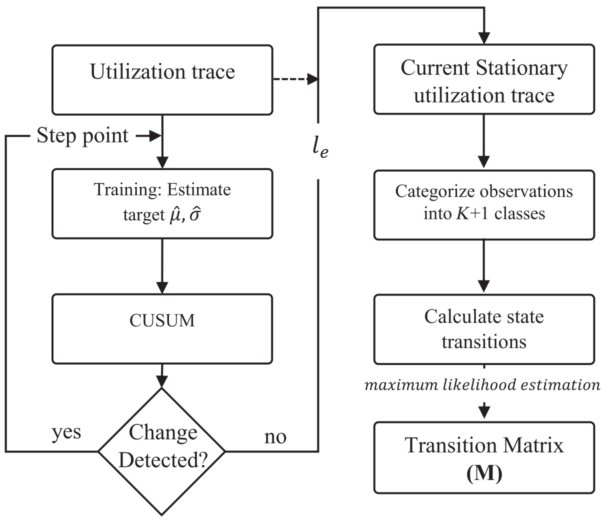
Utilization trace
Step point
Training: Estimate target $\hat{\mu}, \hat{\sigma}$
CUSUM
yes
Change Detected?
no
l_e
Current Stationary utilization trace
Categorize observations into K+1 classes
Calculate state transitions
maximum likelihood estimation
Transition Matrix (M)

FIGURE 2 The high-level view of the online transition matrix estimator algorithm. CUSUM, cumulative sum control chart

length (shown by l_e), and the last l_e time steps form the current stationary utilization trace. The observations of this part of the workload (ie, last l_e samples of the input utilization trace) are categorized into $k + 1$ classes, transitions between each pair of states are counted, and then the transition probabilities of the Markov chain of CPU utilization are calculated by the means of the maximum likelihood estimation method. The advantage of this approach in comparison with the sliding window techniques is that the OTEST algorithm does not select one of the predefined window sizes, and it dynamically determines the largest length of history traces during which the workload is supposed to be stationary.

3.3 | Energy consumption modeling

Power consumption modeling in cloud data centers is an open research agenda.[40-42] Power consumption in a cloud data center is mainly determined by the CPU, memory, disk storage, power supplies, and cooling systems. Showing the power consumed by a physical host h at a given time t ($\in$ [0, T]) with $P_h(t)$, the overall energy consumption of the host in the time interval [0,T] is given by Equation (12).

$$E_h = \int_0^T P_h(t)dt \tag{12}$$

There are different power usage models proposed in the literature to approximate $P_h(t)$. [40-43] The evaluation results have shown that all of their proposed models have acceptable accuracy. Recent studies have discussed that there is a strong and linear relationship between the energy consumption of a host and its CPU utilization even when dynamic voltage and frequency scaling (DVFS) is applied.[20,43] This relation can be modeled as a linear equation as follows.

$$P(u(t)) = P_{\text{idle}} + (P_{\text{busy}} - P_{\text{idle}}) \quad u(t), \tag{13}$$

where P_{idle} is the power usage of a server in the idle state, $u(t)$ is the CPU utilization at time t, and P_{busy} is the power usage of a fully loaded CPU. Fan et al[43] have also proposed an empirical nonlinear model equation as follows:

$$P(u(t)) = P_{\text{idle}} + (P_{\text{busy}} - P_{\text{idle}}) \quad (2u(t) - u(t)^r), \tag{14}$$

where r is a calibration parameter that minimizes the square error and has to be obtained experimentally. The experiments on several thousand nodes with different types of workloads (see Figure 3) indicated that their proposed models predict power consumption accurately (the results in Figure 3 are based on the calibration parameter $r = 1.4$). These precise results can be explained by the fact that voltage and performance scaling is not applied to other system components, such as memory and network interfaces.[20] Therefore, CPU utilization is considered the major resource that contributes to energy consumption in this paper.

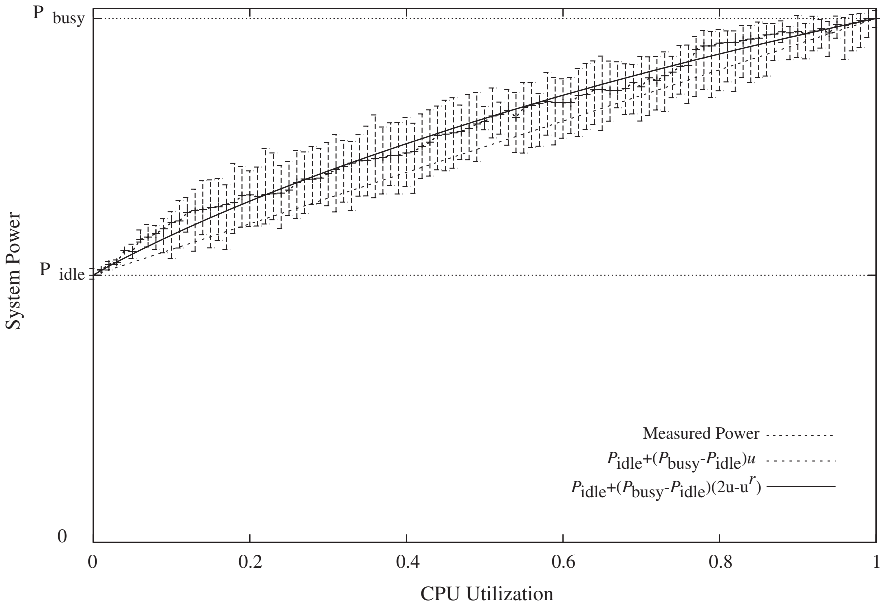

FIGURE 3 The relation between power consumption and CPU utilization of a server[43]

4 | AN ENERGY-EFFICIENT AND QOS-AWARE BFD (EQBFD) ALGORITHM FOR VM PLACEMENT

None of the VM placement algorithms proposed in the literature, which work with unknown nonstationary workloads, are able to consider a certain level of QoS when performing VM placements. In this section, a novel dynamic VM placement algorithm is proposed, which aims at reducing energy consumption while considering QoS objective when making placement decisions. Therefore, this algorithm follows two main goals: (1) satisfying QoS objective and (2) reducing energy consumption. At first, the restriction that a VM placement algorithm should consider to satisfy QoS objective is analyzed, then the EQBFD algorithm is presented.

4.1 | QoS-aware VM placement

The goal of this section is to provide an approach that can be used to consider QoS objective when performing VM placements. For this purpose, it is needed to analyze the required condition on a PM to make sure that the machine is appropriate for hosting a certain VM based on the QoS objective.

4.1.1 | QoS-aware VM placement constraints

It can be shown that the Markov process of CPU utilization on a physical host can be assumed to be ergodic. In general, according to the observed CPU utilization of the VMs residing on a host, the total requested utilization on each host can take any value in the interval $\left[0, \max\left(d_u\right)\right]$; thus, $\hat{p}_{ij} > 0$. In addition, the number of states is finite (ie, $|S| = k+1$), therefore:

1. If starting from state s_i, return to the state s_i is certain, ie, all states are recurrent.
2. There is a path from any state of utilization to any other state, ie, the Markov chain is irreducible.
3. All states are aperiodic.

In a finite-state Markov chain, all recurrent states are positive recurrent.[44] From this, it can be concluded that the aperiodic Markov chain of utilization on each PM is ergodic. For an irreducible ergodic Markov chain $\lim_{t\to\infty} p_{ij}^t$ exists and is independent of i[44]. Furthermore, letting

$$P_{s_j} = \lim_{t\to\infty} p_{ij}^t, \quad j \geq 0 \tag{15}$$

then P_{s_j} is the unique nonnegative solution of

$$P_{s_j} = \sum_{i=1}^{k} P_{s_i} p_{ij}, \quad j \geq 0, \sum_{j=1}^{k} P_{s_j} = 1 \tag{16}$$

The above statements mean that P_{s_j} is independent of the initial state i and it is the limiting probability that utilization process will be in state j at time t. In other words, the equilibrium distribution vector $P_s = [P_{s_0}, P_{s_1}, \quad \dots, P_{s_k}]$ exists, which is unique and independent of the initial state. There are various techniques for finding equilibrium distribution vector.[45] It can be shown that P_{s_j} also equals the long run proportion of time that the process will be in state j.[44] Therefore, the following equation holds for the Markov chain of utilization process on a host h:

$$P^h_{s_k} = \lim_{t\to\infty} \frac{\int_0^t o_h(t)dt}{\int_0^t a_h(t)dt}. \tag{17}$$

It can be concluded that to reach the QoS objective as defined in Equation (2), a QoS-aware placement algorithm should place VMs on PMs in such a way that the following condition holds after placements:

$$\forall h \in \text{PM}s : \text{OTF}_h = \frac{\int_0^t o_h(t)dt}{\int_0^t a_h(t)dt} \approx P^h_{s_k} < \rho, \tag{18}$$

where $P^h_{s_k}$ can be estimated by calculating equilibrium distribution.

4.1.2 | Qualified PMs for QoS-aware VM placement

The above analysis can be used directly in environments with stationary workloads. However, as stated before, real workload data considered in this study are not stationary; thus, the above analysis should be extended to deal with unknown nonstationary workloads. Let $t_s^{(h,i)}$ be the time during which the workload on a host h (as a potential destination of a migrating VM i) will remain stationary after placement and $P^{(h,i)}_{s_k}$ be the long run proportion of time that the workload (including the VM i) on the host h will be in state s_k; therefore, according to the QoS objective, the host h is a *QPM* for hosting the VM i(ie, $h \in QPMs(i)$) if the following condition holds:

$$\frac{t_o^h + t_s^{(h,i)} P^{(h,i)}_{s_k}}{t_a^h + t_s^{(h,i)}} \leq \rho, \tag{19}$$

where $t_o^h = \int_0^t o(h)dt$ and $t_a^h = \int_0^t a(h)dt$. $t_s^{(h,i)}$ is unknown a priori; thus, the VM placement algorithm can control the QoS requirements only by choosing a suitable PM on which the estimated value of $P^{(h,i)}_{s_k}$ (shown with $\hat{P}^{(h,i)}_{s_k}$) is small enough, so that the condition of Equation (19) holds. In this regard, Equation (19) can be rewritten for $\hat{P}^{(h,i)}_{s_k}$ as follows:

$$\hat{P}^{(h,i)}_{s_k} \leq \frac{\rho\left(t_a^h + t_s^{(h,i)}\right)}{t_s^{(h,i)}} - \frac{t_o^h}{t_s^{(h,i)}}. \tag{20}$$

The value of t_o^h will be at most ρt_a^h if a proper QoS assurance policy is used and overloaded PMs are excluded from the list of potential destinations of a VM placement. Thus,

$$\begin{aligned} \frac{\rho t_a^h + \rho t_s^{(h,i)} - \rho t_a^h}{t_s^{(h,i)}} &\leq \frac{\rho\left(t_a^h + t_s^{(h,i)}\right)}{t_s^{(h,i)}} - \frac{t_o^h}{t_s^{(h,i)}} \\ \Rightarrow \rho &\leq \frac{\rho\left(t_a^h + t_s^{(h,i)}\right)}{t_s^{(h,i)}} - \frac{t_o^h}{t_s^{(h,i)}}. \end{aligned} \tag{21}$$

It can be concluded that

$$\hat{P}_{s_k}^{(h,i)} \leq \rho \Rightarrow h \in \text{QPM}s(i). \tag{22}$$

Therefore, a host h with $\hat{P}_{s_k}^{(h,i)} \leq \rho$ is a member of QPM set of VM i, and it will satisfy the QoS objective. Algorithm 1 shows the procedure of finding QPMs for the VM i. The algorithm simulates the utilization trace of after placement on each PM (*simulateUtilizationTraceAfterAllocation*), and the OTEST algorithm is employed to estimate the transition matrix (**M**) according to the simulated trace. By conducting the algorithm completely, the algorithm finds the list of QPMs for a given VM, according to the above statements.

Algorithm 1 The algorithm of finding the list of QPMs for a given VM

```
input  : hostList, overloadedHostList, $vm_i$
output: QPMs for $vm_i$
1  Function findQPMs (hostList, overloadedHostList, $vm_i$) begin
2      qpmList←NULL
3      hostList.remove(overloadedHostList)
4      foreach host h in hostList do
5          $u_{new} \leftarrow simulateUtilizationTraceAfterAllocation(h, vm_i)$ //simulating utilization before performing actual
             migration
6          $\mathbf{M} \leftarrow OTEST(u_{new})$
7          $\hat{P}_{s_k} \leftarrow solve\ (lim_{t\to\infty}\ \mathbf{M}^t)$
8          if $\hat{P}_{s_k} < \rho$ then
9              qpmList.add($h$)
10         end
11     end
12     return qpmList
13 end
```

4.2 | Energy-efficient VM placement

One of the main objectives of VM placement algorithms is reducing total energy consumption. Usually, this is done by selecting a minimum set of PMs to host the VMs. In this regard, we use BFD policy in EQBFD to heuristically reduce the number of used PMs, and it is extended to consider the efficiency of the hosts. We define the efficiency of a host as the ratio of computation power to energy consumption when using DVFS.

$$\eta = \frac{\sum_{i=1}^{|l|} \frac{\bar{c}_i}{w_i}}{|l|}, \tag{23}$$

where $\bar{c}_i$ is the mean CPU capacity in MIPS, w_i is the power consumption in Watts at different load levels, and $|l|$ is the number of states that can be set to the frequency and voltage of a CPU. PMs with larger values of η are recognized as more efficient.

4.3 | EQBFD algorithm

We proposed EQBFD algorithm (Algorithm 2) based on the explanations given in Section 4.1 and Section 4.2. EQBFD selects the most energy-efficient QPMs for the input VMs. To do this, the algorithm finds the list of QPMs for each input VM, and then the most efficient QPM is selected (according to Equation (23)) as the new host of the VM. If there are two or more QPMs with the same value of η, the one with the least AC has more propriety is considered the best fitting QPM.

Algorithm 2 Energy-efficient and QoS-aware VM Placement (EQBFD) algorithm

```
input  : hostList, vmList
output: allocation of VMs
Function EQBFD (hostList, vmList) begin
    vmList.sortDecreasingAverageUtilization()
    overloadedHostList← getOverUtilizedHosts(hostList)
    foreach VM $vm_i$ in vmList do
        maxEfficiency←0
        minAvailableCapacity← $\infty$
        allocatedHost←NULL
        qpmList←findQPMs(hostList, overloadedHostList, $vm_i$)
        foreach host h in qpmList do
            $\eta$ ← getEfficiency(h) //efficiency is calculated based on Equation (23)
            $\phi$ ← getAvailableCapacity(h)
            if maxEfficiency < $\eta$ then
                allocatedHost← h
                maxEfficiency← $\eta$
            end
            else if maxEfficiency==$\eta$ && minAvailableCapacity> $\phi$ then
                allocatedHost← h
                minAvailableCapacity ← $\phi$
            end
        end
        if allocatedHost ≠ NULL then
            allocation.add($vm_i$, allocatedHost)
        end
    end
    return allocation
end
```

5 | QOS ASSURANCE BY DETECTING OVERLOADED HOSTS

As stated before, one of the main goals of this study is establishing QoS guarantees in the server consolidation problem. It was shown in Section 4.1 that the value of $\hat{P}_{s_k}$ should be kept below the desired OTF value. However, when working with real workload data, QoS may be violated due to (1) estimation errors and (2) nonstationary characteristics of the workloads. As the size of estimation trace length (l_e) grows, the estimation of transition probabilities converges to their real values, and thus the estimation error reduces; however, the maximum precision of the estimation is limited to $1/l_e$. Whenever the QoS objective is violated on a host, it is considered to be overloaded, and some VMs should be migrated out. Here, three different policies are suggested to determine the migration time from an overloaded host:

1. ***Deferred***: According to this policy, whenever the value of OTF on an overloaded host exceeds the agreed level of QoS (ie, $OTF > \rho$), then some VMs should be migrated out from the host.
2. ***Prediction-based***: It is usually desired to predict the QoS violation and prevent it by migrating some VMs before the violation takes place. Therefore, based on the Markov chain of utilization on each PM and the transition probabilities during the last l_e observations, if it is predicted that the host will be overloaded in the next time interval (ie, $P(X_{t+1} = s_k|X_t = s_t) > P(X_{t+1} = s_i|X_t = s_t, \forall i : 0 \leq i < k)$, some VMs should be migrated out if the following condition holds:

$$\frac{t_o + t_m + t_i}{t_a + t_m + t_i} > \rho, \tag{24}$$

where t_m is the VM migration time, and t_i is the time interval between subsequent invocations of the VM placement algorithm. If Equation (24) holds, some VMs should be migrated out according to this policy.

3. ***Immediate***: In this policy, the equilibrium distribution vector $(\hat{P}_s)$ is estimated for the utilization model from the last detected step point up to the current time (which can be calculated from the transition matrix that is obtained from the OTEST algorithm). As stated by Ross[44] (discussed in Section 4.1.1), the equilibrium distribution can be used to find out the long run fraction of the time that the utilization process will be in each state; therefore, $\hat{P}_{s_k} > \rho$ means that the OTF value will exceed ρ threshold in long run time, and thus QoS will be violated. To avoid this violation, some VMs should be migrated out from the host.

The *deferred* policy is the straightforward approach in practice, which postpones migrations from an overloaded host to when a QoS violation happens. By predicting QoS violations before their occurrence, QoS violations can be prevented when using the *prediction-based* policy. In the *immediate* policy, some VMs are migrated out from a host as soon as the estimation of the current stationary workload on the host shows that there will be QoS violations in the long run. Both the *immediate* and *prediction-based* policies operate like the *deferred* policy if an actual SLA violation happens in the system. The *immediate* policy is pessimistic while the *deferred* policy is the most optimistic which let VMs work on a host hoping that all will go well in future.

6 | OPTIMIZING ENERGY CONSUMPTION BY DETERMINING UNDERLOADED HOSTS

It is important to find underloaded hosts to decrease the rate of the cost to efficiency. We consider two conditions to specify a low-loaded host. A host is a candidate to be underloaded (1) when the average utilization during the last l_e observations is less than 50% of its capacity or (2) when C_t^- in the standardized CUSUM signals. If C_t^- signals when applying CUSUM to the workload on a PM, it can be concluded that the process mean has decreased significantly. Note that the target mean and variance of the CUSUM algorithm are set according to the utilization process of the last time that a PM had been the source or the destination of a VM migration.

To determine underloaded PMs, the list of candidate machines is sorted by CPU utilization ascending, then iteratively for each PM, the data center manager calls the EQBFD algorithm and attempts to place all the VMs from the host on other hosts while keeping them not overloaded. If such a placement is feasible, the VMs are set for migration to the determined target hosts.

7 | SIMULATION AND RESULT ANALYSIS

It is extremely difficult to conduct repeatable large-scale experiments and evaluation of proposed algorithms on real systems due to many limitations such as expense, and the effects of experiments on system's availability, security, and risks. To ensure the repeatability of the experiments we set up the experimental environment using an extension of CloudSim toolkit.[46] It is a simulation tool which has been developed by the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne. For performance evaluation of the proposed system, a data center with 800 heterogeneous servers with real server configurations is simulated: 400 HP ProLiant ML110 G5 (Intel Xeon 3075, 2 cores x 2660 MHz, 4 GB), and 400 HP ProLiant ML110 G4 (Intel Xeon 3040, 2 cores x 1860 MHz, 4 GB). Power consumptions of the PMs are computed based on the data provided in CloudSim toolkit. There are also four types of VMs inspired form Amazon EC2 instance types with the exception that all the VMs are considered as single-core instances because the workload traces that are used for the simulations come from single-core VMs. For the same reason, the amount of RAM is divided by the cores for each VM type. The specifications of the VM types that are used in the simulations are shown in Table 1.

TABLE 1 Four virtual machine (VM) types (Amazon EC2 VM types) used in the simulations

VM Type	CPU (MIPS)	RAM (GB)
High-CPU medium instance	2500	0.85
Extra-large instance	2000	3.75
Small instance	1000	1.7
Micro instance	500	0.613

TABLE 2 Workload data characteristics[18]

Date	Number of VMs	Mean (%)	SD(%)
03/03/2011	1052	12.31	17.09
06/03/2011	898	11.44	16.38
09/03/2011	1061	10.70	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

Abbreviations: VM, virtual machine.

TABLE 3 Summary of simulation settings

Subject	Parameter	Value
Data center	*Number of PMs*	800
Data center	*PM types*	2
Data center	*VM types*	4
Physical machines	*Number of cores*	2
Virtual machines	*Number of cores*	1
Workloads	Total number	11746
Workloads	Duration	24 hours
CUSUM	*Training phase length*	30
CUSUM	*Allowance value (R)*	1.5σ
CUSUM	*Decision interval (H)*	5σ

Abbreviations: CUSUM, cumulative sum control chart; PM, physical machines; VM, virtual machine.

To evaluate the proposed algorithms under a real-world workload, we use the 10 days' data of the real workload traces of the CoMon project, a monitoring infrastructure for PlanetLab.[18] In that project, the CPU utilization data is collected every five minutes from more than a thousand VMs that are located at more than 500 servers around the world. The characteristics of the data for each day are shown in Table 2. During the simulations, all the basic procedures work based on the default CloudSim operations for VM consolidation. For instance, initially, each VM is randomly assigned a workload trace from one of the VMs from the corresponding day and they are allocated based on to the resource requirements defined by VM type; then, VMs utilize resources according to the assigned workload data during the time to enable dynamic consolidations.[20] Table 3 summarizes the simulation settings and the values of the parameters that are used in the experiments.

7.1 | Performance metrics

ESLATAH: An SLA violation happens if the QoS objective is not satisfied on a host. The SLA time per active host (SLATAH) metric[20] cannot take into account the QoS objective. Therefore, we introduce the extended SLATAH (ESLATAH) metric, which is defined as follows:

$$\text{ESLATAH} = \frac{1}{|\text{PM}|}\sum_{h=1}^{|\text{PM}|}\frac{t_{\text{otf}}^{h}}{t_{a}^{h}}, \tag{25}$$

where $|PM|$ is the total number of PMs and t_{otf}^{h} is the total time that the OTF value on the host h has exceeded the agreed level of QoS (ρ).

ECSV: There is a trade-off between energy consumption and SLA violations. For this reason, we use ECSV metric to perform a more comprehensive comparison of different consolidation policies. It represents the simultaneous optimization of energy consumption and SLA violation.

$$\text{ECSV} = \text{ESLATAH} \times \text{Energy} \tag{26}$$

ECSVM: Virtual machine migrations also impose some costs on cloud service providers. Therefore, similar to some works in the literature,[1,19] to assess the simultaneous minimization of energy, SLA violation, and the number of VMs' migrations, we use a multiparameter metric named ECSVM (energy consumption-SLA violation-migration's count), which is defined as follows:

$$\text{ECSV}M = \text{ECSV} \times \text{Migration} \quad \text{count} \tag{27}$$

7.1.1 | Performance of the proposed algorithms

In this section, we have analyzed the performance of the proposed algorithms in detail, and the results of the suggested policies are compared with the benchmark algorithms in the next section. We run EQBFD algorithm with three different ρ values for OTF as QoS objectives ($\rho = 0.1, 0.2, 0.3$) to realize its impact on the performance metrics. Ten experiments are executed separately for the 10 days of the workloads depicted in Table 2. Note that, in all the simulations, we used a minimum migration time[20] policy as the VM selection policy from overloaded hosts.

The OTF results show that our proposed VM consolidation approach manages to satisfy the specified levels of QoS successfully. As it can be seen in Figure 4, the QoS goals are almost reached when using the *immediate* and *prediction-based* policies as overload detection policies. Figure 5 shows the total energy consumption; results indicate that higher values of ρ lead to lower energy consumption. We expected this because EQBFD algorithm produces denser placements when we set lower QoS performances.

A good overload detection policy will reduce SLA violation in the system (ESLATAH metric). It can be seen in Figure 6 that the *deferred* overload detection policy has the worst ESLATAH values and the *immediate* policy has the best results for the different levels of QoS objectives. This is because the *deferred* policy delays migrations up to the time that an SLA violation happens. In contrast, the *prediction-based* and *immediate* policies try to start migrations before they take place; therefore, they lead to fewer SLA violations. For the same reason, it can be seen in Figure 7 that using the *deferred* policy leads to slightly fewer overload declarations by the overload detection algorithm (we refer to it as *overload detections*) since it claims an overload condition when an actual SLA violation happens. However, the results are close in this metric; thus, no significant difference is observed between the various overload detection policies in the number of migrations as shown in Figure 8.

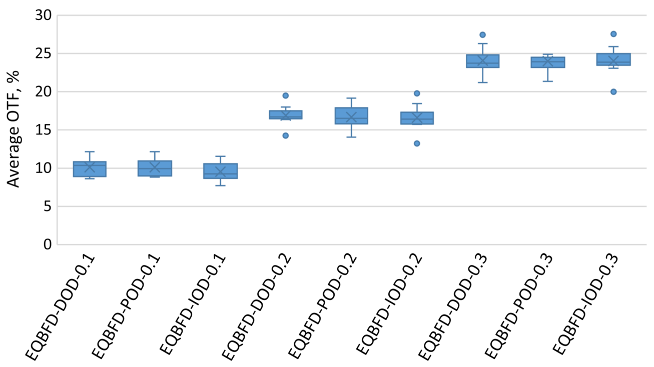

FIGURE 4 Average overload time fraction (OTF) for proposed virtual machine consolidation policies. EQBFD, energy-efficient and quality of service–aware best-fit decreasing [Colour figure can be viewed at wileyonlinelibrary.com]

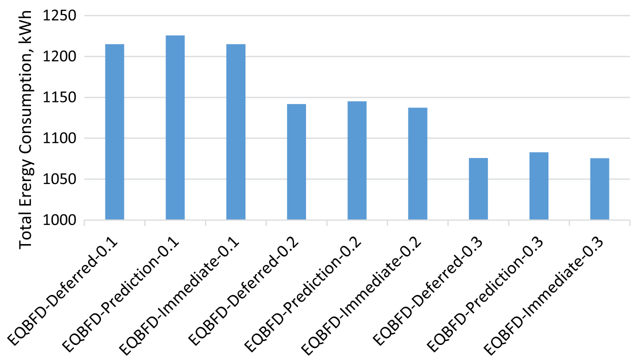

FIGURE 5 Total energy consumption for proposed virtual machine consolidation policies. EQBFD, energy-efficient and quality of service–aware best-fit decreasing [Colour figure can be viewed at wileyonlinelibrary.com]

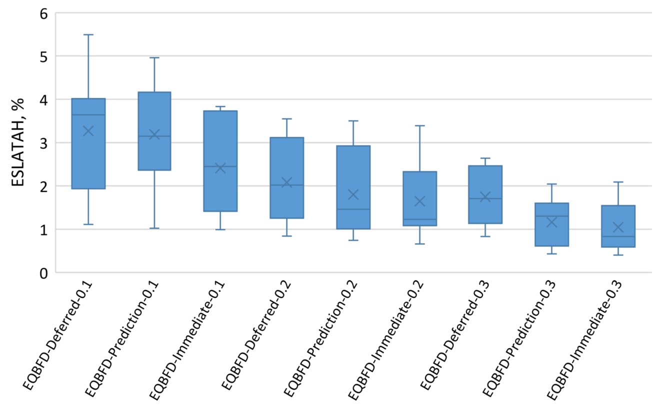

FIGURE 6 Extended service-level agreement time per active host (ESLATAH) for proposed virtual machine consolidation policies. EQBFD, energy-efficient and quality of service–aware best-fit decreasing [Colour figure can be viewed at wileyonlinelibrary.com]

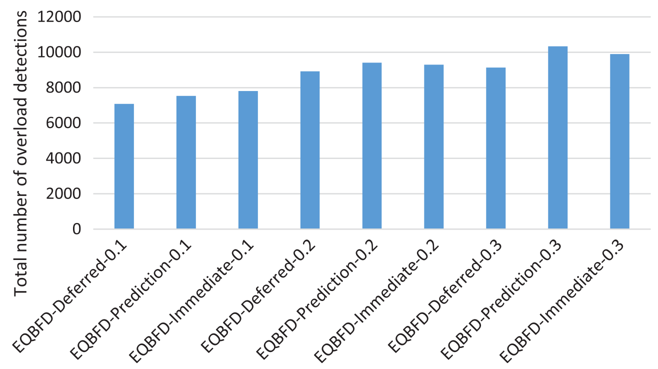

FIGURE 7 Total number of overloads for proposed virtual machine consolidation policies. EQBFD, energy-efficient and quality of service–aware best-fit decreasing [Colour figure can be viewed at wileyonlinelibrary.com]

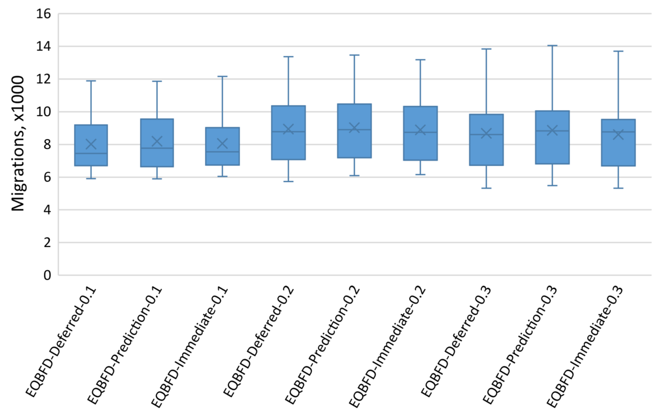

FIGURE 8 Number of migrations for proposed virtual machine consolidation policies. EQBFD, energy-efficient and quality of service–aware best-fit decreasing [Colour figure can be viewed at wileyonlinelibrary.com]

To find out the accuracy of the proposed *prediction-based* overload detection approach, we executed the simulations with the *deferred* method (to specify actual SLA violations) and examined the ability of the proposed *prediction-based* approach in predicting overloads one step beforehand. If the prediction results of this approach be classified as shown in Table 4, then its accuracy[47] is 45% and its precision[47] is 87%. The low accuracy can be justified by the fact that the proposed *prediction-based* method assumes that a host will be oversubscribed in the next time step if the probability of moving to overload state is higher than the probabilities of moving to other states, but it does not consider the magnitude of the probability. However, one may consider a low threshold for overload probability or define other constraints for overload prediction; this subject can be investigated in a separate study.

TABLE 4 Classification of *prediction-based* policy's results

		Predicted	
		Overload	**No overload**
Actual	Overload	True positive	False negative
	No overload	False positive	True negative

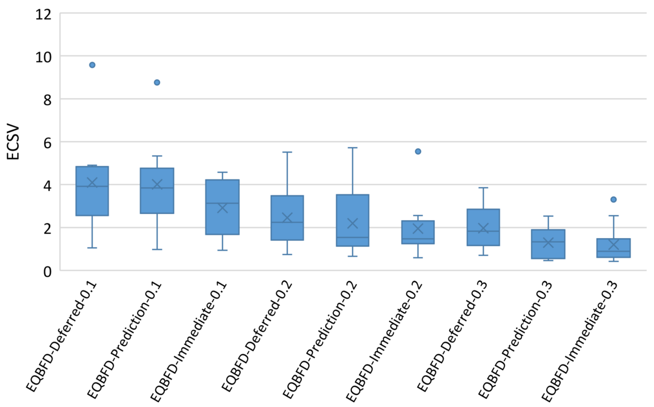

FIGURE 9 Energy consumption and service-level agreement violation metric for proposed virtual machine consolidation policies. EQBFD, energy-efficient and quality of service–aware best-fit decreasing [Colour figure can be viewed at wileyonlinelibrary.com]

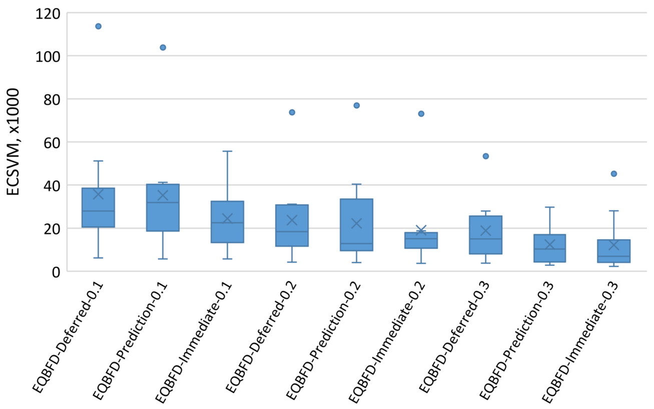

FIGURE 10 Energy consumption, service-level agreement violations, and the number of migrations for proposed virtual machine consolidation policies. EQBFD, energy-efficient and quality of service–aware best-fit decreasing [Colour figure can be viewed at wileyonlinelibrary.com]

Figure 9 shows the results for ECSV metric. The results indicate that the *immediate* policy has a better performance in both energy consumption and SLA violation than the other proposed overload detection policies. Figure 10 shows the result for ECSVM metric, and Table 5 compares our proposed policies containing all our proposed algorithms for different levels of QoS objective the mean values of the ECSVM metric along with the 95% confidence intervals. From the observed results, it can be concluded that the total costs in the system we consider in this study (includes energy consumption, SLA violation, and the number of migrations) reduces as the value of the ρ increases. This is because the costs in the system reduce when the delivered QoS reduces. The results also indicate that the *immediate* overload detection policy has the best, and the *deferred* policy has the worst ECSVM values.

In order to show how well the OTEST algorithm works, we generated an artificial nonstationary workload. This workload is made up of three stationary Markov processes (generated by *hmmgenerate* function of MATLAB language),

TABLE 5 Comparison of the proposed policies regarding the mean values of energy consumption, service-level agreement violations, and number of migrations (ECSVM) metric and 95% confidence intervals

Policy	ECSVM ($x10^3$)	95% CI ($x10^3$)
EQBFD-Deferred-0.1	35.69	(15.41, 55.97)
EQBFD-Prediction-0.1	35.23	(17.18, 53.28)
EQBFD-Immediate-0.1	24.46	(14.64, 34.28)
EQBFD-Deferred-0.2	24.22	(10.69, 37.75)
EQBFD-Prediction-0.2	22.61	(7.27, 37.95)
EQBFD-Immediate-0.2	19.16	(5.89, 32.43)
EQBFD-Deferred-0.3	18.88	(8.93, 28.83)
EQBFD-Prediction-0.3	12.36	(5.77, 18.95)
EQBFD-Immediate-0.3	12.20	(2.87, 21.53)

Abbreviations: EQBFD, energy-efficient and quality of service–aware best-fit decreasing.

TABLE 6 The artificial nonstationary workload transition probabilities

	0:00 - 7:00	7:00 - 17:00	17:00 - 24:00
p_{00}	0.8	0.2	0.8
p_{01}	0.2	0.8	0.2
p_{10}	0.8	0.2	0.8
p_{11}	0.2	0.8	0.2

whose transition probabilities are given in Table 6. State 0 is the host state at normal load, and state 1 is when the host is oversubscribed. We simulated this part of the experiments using the MATLAB language. The performance of the OTEST algorithm in estimating transition probabilities is compared with the sliding window method. Three different sizes for sliding windows are considered: 1 hour (w = small), 3 hours (w = medium), and 5 hours (w = large). The definitions of the scales like *small*, *medium*, and *large* depend on the total length of a workload and the duration of the time that it remains stationary; here, they are only some samples defined according to the artificially generated workload to illustrate the sliding window method performance. As shown in Figure 11, the results imply that OTEST has a high quality of estimation, and it quickly adapts to workload variations. In contrast, the sliding window method with small window size has a low quality of estimation when the workload is stationary for a long time; using larger window sizes also lead to low-performance estimation when there is a switch from a stationary workload to a new stationary workload.

7.2 | Markov-based consolidation policy compared with benchmark approaches

In this section, we compare our proposed Markov-based consolidation solution with various heuristic approaches proposed in the literature.[1,10,20,21,29,30] The Markov-based approach in this section includes EQBFD policy for VM placement, the *immediate* policy for overload detection, and the underload detection policy explained in Section 6. A three-segmented naming format is used here for the notation of the different benchmark solutions assessed in this section. Different sections of the naming format are arranged according to the policy adopted for VM placement, the policy adopted for overload detection, and the policy for underload detection. The notations are constructed by connecting the abbreviation of the policies used for each section using slash lines.

The benchmark solutions consist of support vector regression (SVR),[10] local regression (LR)[20], median absolute deviation (MAD),[20] and interquartile range (IQR)[20] for detecting overloaded PMs; PABFD,[20] AVL,[29] MFPED,[30] TPSA,[1] UMC,[21] and SBBFD[10] are selected for VM placements; TACND,[1] MDL,[1] AC,[1] VDT,[21] stochastic-based (SB),[10] and simple method (SM)[20] are selected for detecting underloaded PMs.

It can be seen that the Markov-based consolidation approach has led to a reduction in total energy consumption (up to 15% in comparison with the best of the benchmark policies in this metric, ie, AVL-LR-SM) as shown in Figure 12. This implies that the proposed energy-efficient placement method used along with the QoS-aware placement method

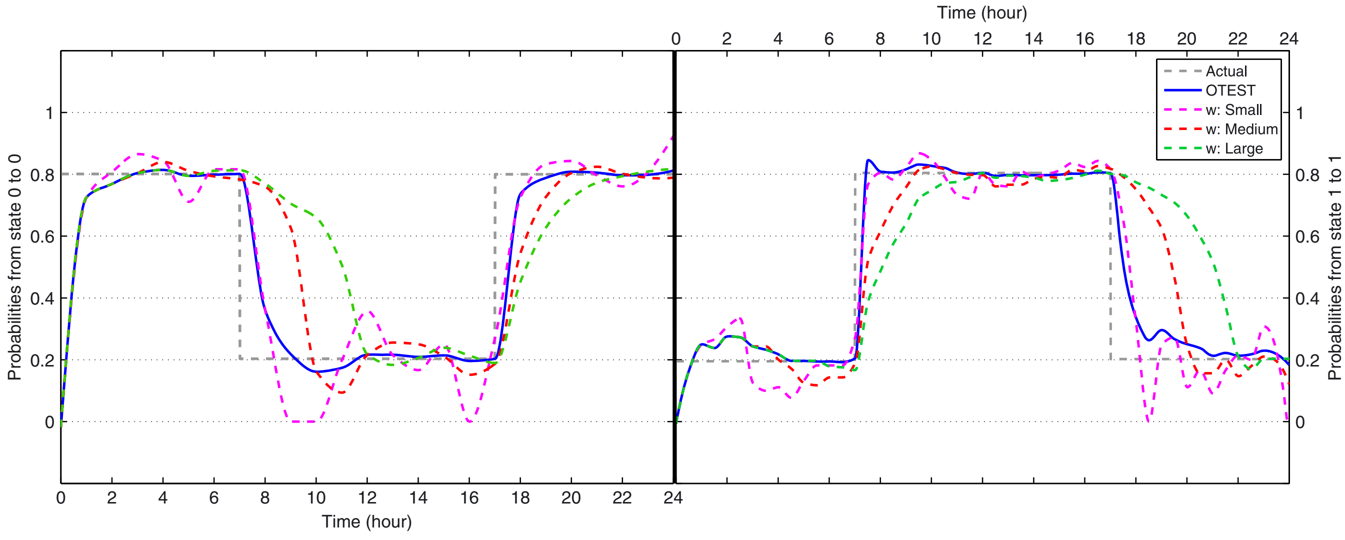

FIGURE 11 The estimated values of transition probabilities compared to actual values of transition probabilities [Colour figure can be viewed at wileyonlinelibrary.com]

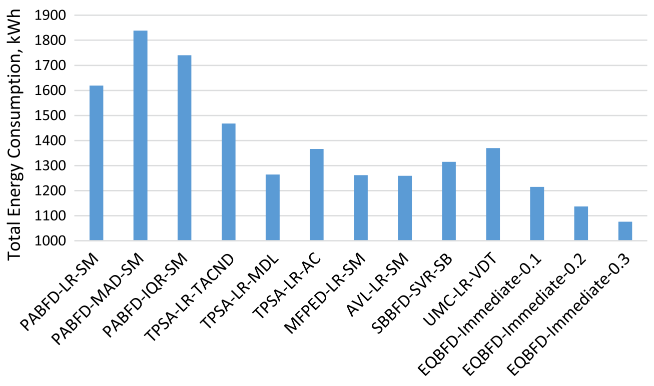

FIGURE 12 Energy consumption for combination of different policies for virtual machine (VM) consolidation process. AVL, availability-based VM allocation; EQBFD, energy-efficient and quality of service–aware best-fit decreasing; IQR, interquartile range; LR, local regression; MAD, median absolute deviation; MDL, migration delay; MFPED, medium-fit power-efficient decreasing; PABFD, power-aware best fit decreasing; SBBFD, stochastic process–based best fit decreasing; SM, simple method; TPSA, TOPSIS power and SLA–aware allocation; TACND, TOPSIS AC, number of virtual machines, and MDL; UMC, utilization and minimum correlation; VDL, virtual machine-based dynamic threshold [Colour figure can be viewed at wileyonlinelibrary.com]

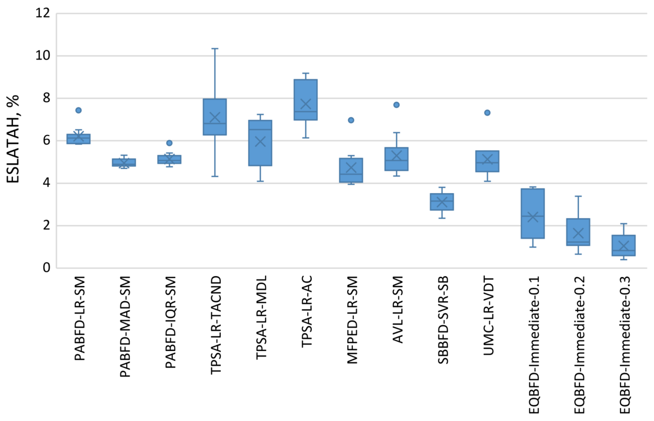

FIGURE 13 Extended service-level per active host (ESLATAH) for combination of different policies for virtual machine (VM) consolidation process. AVL, availability-based VM allocation; EQBFD, energy-efficient and quality of service–aware best-fit decreasing; IQR, interquartile range; LR, local regression; MAD, median absolute deviation; MDL, migration delay; MFPED, medium-fit power-efficient decreasing; PABFD, power-aware best fit decreasing; SBBFD, stochastic process–based best fit decreasing; SM, simple method; TPSA, TOPSIS power and SLA–aware allocation; TACND, TOPSIS AC, number of VMs, and MDL; UMC, utilization and minimum correlation; VDL, VM-based dynamic threshold [Colour figure can be viewed at wileyonlinelibrary.com]

reduces energy consumption more than the benchmark algorithms. The explained method for QoS-aware VM placement leads to more energy efficient placements since it allows some VM placements which are considered as overload condition in other placement algorithms; therefore, this kind of placements are not allowed in the benchmark VM placement algorithm (ie, PABFD, TPSA, MFPED, AVL, UMC, SBBFD).

Figure 13 shows that our proposed consolidation policy reduces the *ESLATAH* metric more than the other consolidation policies. This shows the efficiency of the *immediate* overload detection policy in preventing SLA violations, in addition to the fact that the benchmark overload detection policies are incapable of satisfying a desired level of QoS.

Figure 14 shows that the total number of *overload detections* in our proposed system has considerably reduced, which is up to 58% reduction in comparison with the fewest of the benchmark policies, ie, SBBFD-SVR-SB. The main reason is that all of the suggested overload detection policies in this paper allow late overload condition declarations according to the QoS objective; in contrast, the benchmark solutions (ie, SVR, LR, MAD, and IQR) recognize an overload condition as soon as they predict a resource shortage; therefore, there are fewer overloaded server declarations in our proposed policy. The number of migrations is given in Figure 15; results show that our proposed consolidation

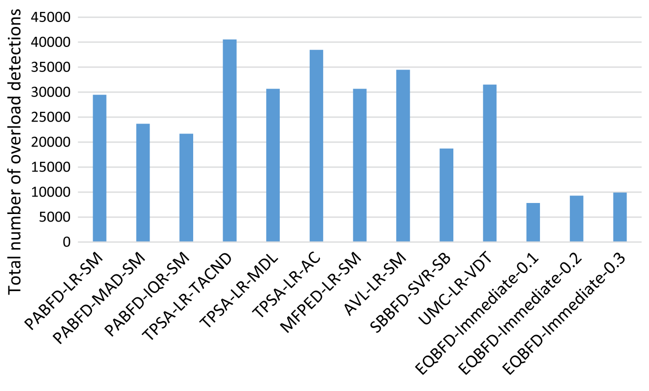

FIGURE 14 Total number of overloads for combination of different policies for virtual machine (VM) consolidation process. AVL, availability-based VM allocation; ECSVM, energy consumption, service-level agreement violations, and the number of migrations; EQBFD, energy-efficient and quality of service–aware best-fit decreasing; IQR, interquartile range; LR, local regression; MAD, median absolute deviation; MDL, migration delay; MFPED, medium-fit power-efficient decreasing; PABFD, power-aware best fit decreasing; SBBFD, stochastic process–based best fit decreasing; SM, simple method; TPSA, TOPSIS power and SLA–aware allocation; TACND, TOPSIS AC, number of VMs, and MDL; UMC, utilization and minimum correlation; VDL, VM-based dynamic threshold [Colour figure can be viewed at wileyonlinelibrary.com]

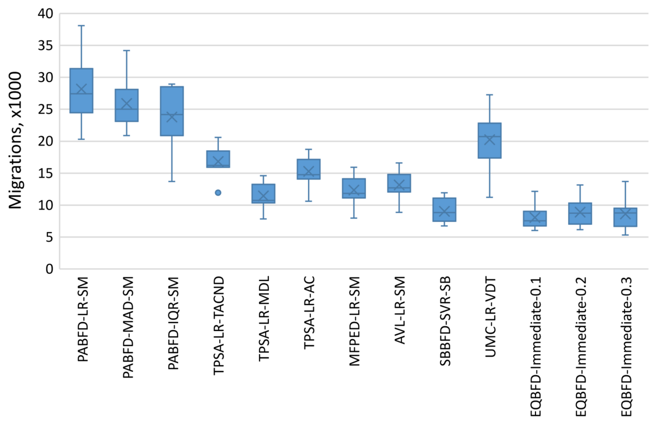

FIGURE 15 Number of migrations for combination of different policies for virtual machine (VM) consolidation process. AVL, availability-based VM allocation; ECSVM, energy consumption, service-level agreement violations, and the number of migrations; EQBFD, energy-efficient and quality of service–aware best-fit decreasing; IQR, interquartile range; LR, local regression; MAD, median absolute deviation; MDL, migration delay; MFPED, medium-fit power-efficient decreasing; PABFD, power-aware best fit decreasing; SBBFD, stochastic process–based best fit decreasing; SM, simple method; TPSA, TOPSIS power and SLA–aware allocation; TACND, TOPSIS AC, number of VMs, and MDL; UMC, utilization and minimum correlation; VDL, VM-based dynamic threshold [Colour figure can be viewed at wileyonlinelibrary.com]

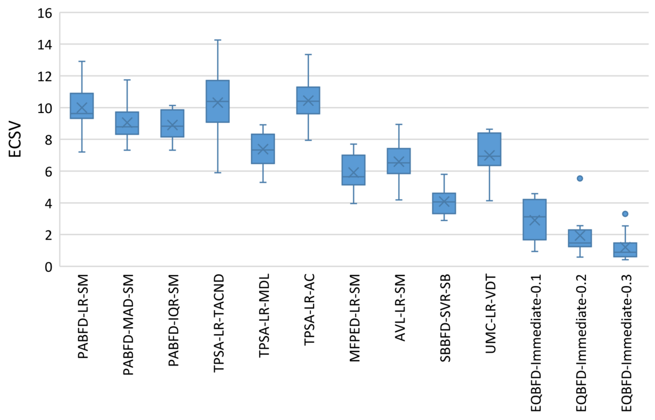

FIGURE 16 Energy consumption and service-level agreement (SLA) violation metric (ECSV) for combination of different policies for virtual machine (VM) consolidation process. AVL, availability-based VM allocation; EQBFD, energy-efficient and quality of service–aware best-fit decreasing; IQR, interquartile range; LR, local regression; MAD, median absolute deviation; MDL, migration delay; MFPED, medium-fit power-efficient decreasing; PABFD, power-aware best fit decreasing; SBBFD, stochastic process–based best fit decreasing; SM, simple method; TPSA, TOPSIS power and SLA–aware allocation; TACND, TOPSIS AC, number of VMs, and MDL; UMC, utilization and minimum correlation; VDL, VM-based dynamic threshold [Colour figure can be viewed at wileyonlinelibrary.com]

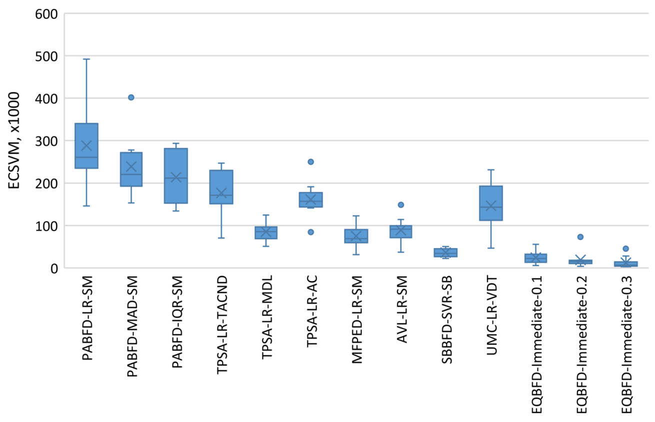

FIGURE 17 Energy consumption, service-level agreement (SLA) violations, and the number of migration (ECSVM) metric for combination of different policies for virtual machine (VM) consolidation process. AVL, availability-based VM allocation; EQBFD, energy-efficient and quality of service–aware best-fit decreasing; IQR, interquartile range; LR, local regression; MAD, median absolute deviation; MDL, migration delay; MFPED, medium-fit power-efficient decreasing; PABFD, power-aware best fit decreasing; SBBFD, stochastic process–based best fit decreasing; SM, simple method; TPSA, TOPSIS power and SLA–aware allocation; TACND, TOPSIS AC, number of virtual machines, and MDL; UMC, utilization and minimum correlation; VDL, virtual machine–based dynamic threshold [Colour figure can be viewed at wileyonlinelibrary.com]

approach is very close to SBBFD-SVR-SB, and it outperforms the other benchmark algorithms in this metric. Finally, to perform comprehensive comparisons between our proposed system and benchmark policies, we employ ECSV and ECSVM metrics. It can be concluded from the results, which are illustrated in Figure 16 and Figure 17, that our proposed system has a better performance in both ESCV and ECSVM metrics for all the values of QoS objective. We have also conducted some paired *t*-tests to specify the significance of the difference between our proposed system and benchmark policies (see Table 7). The *t*-tests results show that our proposed system with different levels of QoS objective leads to a statistically significantly lower value of the ECSVM metric, especially when we set higher values for ρ.

Policy 1 (ECSVM x1000)	Policy 2 (ECSVM x1000)	Difference (x1000)	P-value
EQBFD-Immediate-0.1	PABFD-LR-SM	263.83 (201.89, 325.76)	P-value < 0.001
	PABFD-MAD-SM	214.46 (169.9, 259.02)	P-value < 0.001
	PABFD-IQR-SM	189.08 (143.54, 234.63)	P-value < 0.001
	TPSA-LR-TACND	152.22 (119.46, 184.98)	P-value < 0.001
	TPSA-LR-MDL	60.87 (47.32, 74.42)	P-value < 0.001
	TPSA-LR-AC	136.98 (112.94, 161.01)	P-value < 0.001
	MFPED-LR-SM	50.53 (33.53, 67.53)	P-value < 0.001
	AVL-LR-SM	64.41 (45.73, 159.93)	P-value < 0.001
	SBBFD-SVR-SB	11.64 (1.52, 21.77)	P-value < 0.05
	UMC-LR-VDT	122.16 (84.39, 159.93)	P-value < 0.001
EQBFD-Immediate-0.2	PABFD-LR-SM	269.12 (211.52, 326.73)	P-value < 0.001
	PABFD-MAD-SM	219.75 (180.04, 259.47)	P-value < 0.001
	PABFD-IQR-SM	194.38 (144.97, 243.79)	P-value < 0.001
	TPSA-LR-TACND	157.51 (124.61, 190.41)	P-value < 0.001
	TPSA-LR-MDL	66.17 (51.18, 81.15)	P-value < 0.001
	TPSA-LR-AC	142.27 (120.03, 164.51)	P-value < 0.001
	MFPED-LR-SM	55.83 (39.47, 72.18)	P-value < 0.001
	AVL-LR-SM	69.70 (53.50, 85.91)	P-value < 0.001
	SBBFD-SVR-SB	16.94 (5.86, 28.01)	P-value < 0.01
	UMC-LR-VDT	127.46 (90.59, 164.33)	P-value < 0.001
EQBFD-Immediate-0.3	PABFD-LR-SM	276.08 (216.08, 336.09)	P-value < 0.001
	PABFD-MAD-SM	226.71 (183.89, 269.53)	P-value < 0.001
	PABFD-IQR-SM	201.34 (157.38, 245.3)	P-value < 0.001
	TPSA-LR-TACND	164.47 (132.22, 196.72)	P-value < 0.001
	TPSA-LR-MDL	73.13 (62.16, 84.09)	P-value < 0.001
	TPSA-LR-AC	149.23 (126.22, 172.24)	P-value < 0.001
	MFPED-LR-SM	62.79 (49.73, 75.84)	P-value < 0.001
	AVL-LR-SM	76.66 (61.70, 91.62)	P-value < 0.001
	SBBFD-SVR-SB	23.9 (14.71, 33.09)	P-value < 0.001
	UMC-LR-VDT	134.42 (100.02, 168.81)	P-value < 0.001

Abbreviations: AVL, availability-based virtual machine allocation; EQBFD, energy-efficient and quality of service–aware best-fit decreasing; IQR, interquartile range; LR, local regression; MAD, median absolute deviation; MDL, migration delay; MFPED, medium-fit power-efficient decreasing; PABFD, power-aware best fit decreasing; SBBFD, stochastic process–based best fit decreasing; SM, simple method; TPSA, TOPSIS power and SLA–aware allocation; TACND, TOPSIS AC, number of virtual machines, and MDL; UMC, utilization and minimum correlation; VDL, virtual machine-based dynamic threshold.

TABLE 7 Comparison of the proposed and benchmark virtual machine consolidation policies using paired *t*-tests

8 | CONCLUSION

In this paper, a Markov-based server consolidation approach has been proposed for cloud data centers with QoS constraints. The proposed system has employed an online method to automatically adapt the Markov chain model of CPU utilization on each host to the nonstationary behavior of the workload data. Also, a new energy-efficient and QoS-aware VM placement algorithm have been suggested, which determines the suitable hosts for VMs according to the QoS objective. Therefore, the system is allowed to consider the desired level of QoS when relocating the VMs. To ensure the QoS establishment in the data centers with unknown a priori workloads, three QoS assurance policies have been introduced. We have evaluated the performance of our proposed policies by conducting extensive simulations (The source code of the proposed algorithms is available online[§]); the results have shown that our proposed VM consolidation algorithms outperform the consolidation approaches in the literature in terms of the ECSVM metric because of the reduction in energy consumption, SLA violation and the number of VM migrations. As a part of future work, we plan to consider data centers with geographical expansion and consider energy consumed in switching equipment. Also, other variations of Markov chains, such as hidden Markov models, Poisson hidden Markov models, and two-level Bayesian hidden Markov models can be studied to see if they can be used improve the results of the proposed algorithms in this paper.

[§]https://gitlab.com/monshizade/mbdsc/

ORCID

Esmaeil Zeinali https://orcid.org/0000-0002-7099-5934

REFERENCES

1. Arianyan E, Taheri H, –Sharifian S. Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers. *Comput Electr Eng*. 2015;47:222-240.
2. Rajabzadeh M, Haghighat AT. Energy-aware framework with Markov chain-based parallel simulated annealing algorithm for dynamic management of virtual machines in cloud data centers. *J Supercomput*. 2017;73(5):2001-2017.
3. Rimal BP, Choi E, Lumb I. A taxonomy and survey of cloud computing systems. In: NCM '09 Proceedings of the 2009 Fifth International Joint Conference on INC, IMS and IDC; 2009; Seoul, South Korea.
4. Garg SK, Toosi AN, Gopalaiyengar SK, Buyya R. SLA-based virtual machine management for heterogeneous workloads in a cloud datacenter. *J Netw Comput Appl*. 2014;45:108-120.
5. Abadi RMB, Rahmani AM, Alizadeh SH. Server consolidation techniques in virtualized data centers of cloud environments: a systematic literature review. *Softw Pract Exper*. 2018;48:1688-1726. https://doi.org/10.1002/spe.2582
6. Li H, Zhu G, Zhao Y, Dai Y, Tian W. Energy-efficient and QoS-aware model based resource consolidation in cloud data centers. *Cluster Computing*. 2017;20(3):2793-2803.
7. Manvi SS, Shyam GK. Resource management for infrastructure as a service (IaaS) in cloud computing: a survey. *J Netw Comput Appl*. 2014;41:424-440.
8. Buyya R, Beloglazov A, Abawajy J. Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges. In: Proceedings of the 2010 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2010); 2010; Las Vegas, NV.
9. Beloglazov A, Buyya R. Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. *IEEE Trans Parallel Distrib Syst*. 2013;24(7):1366-1379.
10. Naeen HM, Zeinali E, Haghighat AT. A stochastic process-based server consolidation approach for dynamic workloads in cloud data centers. *J Supercomput*. 2018. https://doi.org/10.1007/s11227-018-2431-5
11. Rao KS, Thilagam PS. Heuristics based server consolidation with residual resource defragmentation in cloud data centers. *Future Gener Comput Syst*. 2015;50:87-98.
12. Farahnakian F, Ashraf A, Pahikkala T, et al. Using ant colony system to consolidate VMs for green cloud computing. *IEEE Trans Serv Comput*. 2015;8(2):187-198.
13. Tang M, Pan S. A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers. *Neural Process Lett*. 2015;41(2):211-221.
14. Speitkamp B, Bichler M. A mathematical programming approach for server consolidation problems in virtualized data centers. *IEEE Trans Serv Comput*. 2010;3(4):266-278.
15. Zhang L, Zhuang Y, Zhu W. Constraint programming based virtual cloud resources allocation model. *Int J Hybrid Inf Technol*. 2013;6(6):333-344.
16. Beloglazov A, Buyya R. OpenStack Neat: a framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds. *Concurr Comput Pract Exp*. 2015;27(5):1310-1333.
17. Feller E, Rilling L, Morin C. Snooze: a scalable and autonomic virtual machine management framework for private clouds. In: Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012); 2012; Ottawa, ON.
18. Park K, Pai VS. CoMon: a mostly-scalable monitoring system for PlanetLab. *ACM SIGOPS Oper Syst Rev*. 2006;40(1):65-74.
19. Arianyan E, Taheri H, Khoshdel V. Novel fuzzy multi objective DVFS-aware consolidation heuristics for energy and SLA efficient resource management in cloud data centers. *J Netw Comput Appl*. 2017;78:43-61.
20. Beloglazov A, Buyya R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurr Comput Pract Exp*. 2012;24(13):1397-1420.
21. Horri A, Mozafari MS, Dastghaibyfard G. Novel resource allocation algorithms to performance and energy efficiency in cloud computing. *J Supercomput*. 2014;69(3):1445-1461.
22. Verma A, Dasgupta G, Nayak TK, De P, Kothari R. Server workload analysis for power minimization using consolidation. In: Proceedings of the 2009 Conference on USENIX Annual Technical Conference; 2009; San Diego, CA.
23. Abadi RMB, Rahmani AM, Alizadeh SH. Self-adaptive architecture for virtual machines consolidation based on probabilistic model evaluation of data centers in Cloud computing. *Cluster Computing*. 2018;21:1711-1733.
24. Zhang S, Qian Z, Luo Z, Wu J, Lu S. Burstiness-aware resource reservation for server consolidation in computing clouds. *IEEE Trans Parallel Distrib Syst*. 2016;27(4):964-977.
25. Lee YC, Zomaya AY. Energy efficient utilization of resources in cloud computing systems. *J Supercomput*. 2012;60(2):268-280.
26. Asyabi E, Sharifi M. A new approach for dynamic virtual machine consolidation in cloud data centers. *Int J Mod Educ Comput Sci*. 2015;7(4):61.
27. Wang Y, Wang X. Performance-controlled server consolidation for virtualized data centers with multi-tier applications. *Sustain Comput: Inform Syst*. 2014;4(1):52-65.

28. Gao Y, Guan H, Qi Z, Song T, Huan F, Liu L. Service level agreement based energy-efficient resource management in cloud data centers. *Comput Electr Eng*. 2014;40(5):1621-1633.
29. Lu K, Yahyapour R, Wieder P, Kotsokalis C, Yaqub E, Jehangiri AI. QoS-aware VM placement in multi-domain service level agreements scenarios. Paper presented at: IEEE Sixth International Conference on Cloud Computing; 2013; Santa Clara, CA.
30. Moges FF, Abebe SL. Energy-aware VM placement algorithms for the OpenStack Neat consolidation framework. *J Cloud Comput*. 2019;8(1):2.
31. Zheng Q, Veeravalli B. Utilization-based pricing for power management and profit optimization in data centers. *J Parallel Distr Com*. 2012;72(1):27-34.
32. Srikantaiah S, Kansal A, Zhao F. Energy aware consolidation for cloud computing. In: Proceedings of the 2008 Conference on Power Aware Computing and Systems; 2008; San Diego, CA.
33. Mars J, Tang L, Hundt R, Skadron K, Soffa ML. Bubble-up: increasing utilization in modern warehouse scale computers via sensible co-locations. In: Proceedings of the 44th annual IEEE/ACM International Symposium on Microarchitecture; 2011; Porto Alegre, Brazil.
34. Govindan S, Liu J, Kansal A, Sivasubramaniam A. Cuanta: quantifying effects of shared on-chip resource interference for consolidated virtual machines. In: Proceedings of the 2nd ACM Symposium on Cloud Computing; 2011; Cascais, Portugal.
35. Chen S, Delimitrou C, Martínez JF. PARTIES: QoS-aware resource partitioning for multiple interactive services. Paper presented at: 2019 Architectural Support for Programming Languages and Operating Systems (ASPLOS); 2019; Providence, RI.
36. Delimitrou C, Kozyrakis C. Quasar: resource-efficient and QoS-aware cluster management. *ACM SIGARCH Comput Archit News*. 2014;42:127-144.
37. Delimitrou C, Kozyrakis C. Paragon: QoS-aware scheduling for heterogeneous datacenters. *ACM SIGPLAN Notices*. 2013;4:77-88.
38. Delimitrou C, Kozyrakis C. QoS-aware scheduling in heterogeneous datacenters with paragon. *ACM Trans Comput Syst*. 2013;31(4). Article No. 12.
39. Montgomery DC. *Introduction to Statistical Quality Control*. 6th ed. New York, NY: Wiley; 2009.
40. Castañé GG, Nuñez A, Llopis P, Carretero J. E-mc^2: a formal framework for energy modelling in cloud computing. *Simul Model Pract Theory*. 2013;39:56-75.
41. Kurowski K, Oleksiak A, Piątek W, Piontek T, Przybyszewski A, Węglarz J. DCworms - a tool for simulation of energy efficiency in distributed computing infrastructures. *Simul Model Pract Theory*. 2013;39:135-151.
42. Jararweh Y, Jarrah M, Alshara Z, Alsaleh MN, Al-Ayyoub M. CloudExp: a comprehensive cloud computing experimental framework. *Simul Model Pract Theory*. 2014;49:180-192.
43. Fan X, Weber WD, Barroso LA. Power provisioning for a warehouse-sized computer. *ACM SIGARCH Comput Archit News*. 2007;2:13-23.
44. Ross SM. *Introduction to Probability Models*. 10th ed. Burlington, MA: Academic Press; 2014.
45. Paige C, Styan GP, Wachter PG. Computation of the stationary distribution of a Markov chain. *J Stat Comput Sim*. 1975;4(3):173-186.
46. Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw Pract Exp*. 2011;41(1):23-50.
47. Galdi P, Tagliaferri R. Data mining: accuracy and error measures for classification and prediction. In: *Encyclopedia of Bioinformatics and Computational Biology*. Amsterdam, The Netherlands: Elsevier; 2018:431-436.

How to cite this article: Monshizadeh Naeen H, Zeinali E, Toroghi Haghighat A. Adaptive Markov-based approach for dynamic virtual machine consolidation in cloud data centers with quality-of-service constraints. *Softw: Pract Exper*. 2020;50:161–183. https://doi.org/10.1002/spe.2764