Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 7848232, 13 pages
https://doi.org/10.1155/2020/7848232

Research Article

Virtual Machine Consolidation with Minimization of Migration Thrashing for Cloud Data Centers

Xialin Liu,[1,2,3] **Junsheng Wu**,[4] **Gang Sha**,[1] **and Shuqin Liu**[2,3]

[1]*School of Computer Science, Northwestern Polytechnical University, Xi'an, Shaanxi, China*
[2]*School of Computer Science and Technology, Xi'an University of Posts and Telecommunications, Xi'an, Shaanxi, China*
[3]*Shaanxi Key Laboratory of Network Data Analysis and Intelligent Processing, Xi'an University of Posts and Telecommunications, Xi'an, Shaanxi, China*
[4]*School of Software and Microelectronics, Northwestern Polytechnical University, Xi'an, China*

Correspondence should be addressed to Xialin Liu; liuxialin@sina.com

Received 26 January 2020; Revised 28 May 2020; Accepted 12 June 2020; Published 3 August 2020

Academic Editor: Ricardo Perera

Cloud data centers consume huge amount of electrical energy bringing about in high operating costs and carbon dioxide emissions. Virtual machine (VM) consolidation utilizes live migration of virtual machines (VMs) to transfer a VM among physical servers in order to improve the utilization of resources and energy efficiency in cloud data centers. Most of the current VM consolidation approaches tend to aggressive-migrate for some types of applications such as large capacity application such as speech recognition, image processing, and decision support systems. These approaches generate a high migration thrashing because VMs are consolidated to servers according to VM's instant resource usage without considering their overall and long-term utilization. The proposed approach, dynamic consolidation with minimization of migration thrashing (DCMMT) which prioritizes VM with high capacity, significantly reduces migration thrashing and the number of migrations to ensure service-level agreement (SLA) since it keeps VMs likely to suffer from migration thrashing in the same physical servers instead of migrating. We have performed experiments using real workload traces compared to existing aggressive-migration-based solutions; through simulations, we show that our approach improves migration thrashing metric by about 28%, number of migrations metric by about 21%, and SLAV metric by about 19%.

1. Introduction

Cloud computing is currently receiving considerable attention from both academic and industrial communities due to the ever-growing applications and the economy of cloud computing. Cloud computing can be considered as a computing paradigm with many exciting features such as on-demand computing resources, elastic scaling, elimination of front-end capital, and operational cost, and delivers infrastructure, platform, and software as services to end users in a pay-as-you-go pattern [1]. Cloud computing data centers consume enormous amount of electrical power resulting in high operating costs and carbon dioxide emissions. The amount of energy consumed by such system is comparable to that of Argentina and continues to grow 11% annually [2]. Google Data Centers are estimated to have consumed 260 million Watts of energy (0.01% of the world's energy) in 2013 [3]. The reason for high energy consumption is the inefficient use of hardware resources. Idle servers can consume about 60% of their peak power [4–6]. One way to deal with energy inefficiencies is to take advantage of the virtualization [7] capabilities that IBM launched in the 1960s as a transparent way to provide concurrent and interactive access to large computers by multiple users [8]. This virtualization technique is the abstraction of underlying physical resources, provides a uniform view of a pool of resources in cloud data centers where a single server can hold more than one computing environments known as *virtual machine*. Advantages brought by virtualization involve higher utilization, isolation, manageability, and reliability of distributed systems.

Live migration of virtual machine (VM) is a major advantage of virtualization, which is a helpful tool to manage cloud environment resources. On one hand, a VM can be

migrated seamlessly and transparently from one physical server to another [9]. On the other hand, live migration provides the ability to move virtual machines (VMs) without stopping them. This process is referred to as VM consolidation, which enables consolidating VM to the minimum physical servers dynamically when user demand changes and VMs are needed to be reallocated resources, and effectively reducing the power consumption of a cloud data center by shutting down unused servers [10]. There are several challenges associated with the VM consolidation problem. The first challenge is associated with the NP-hard nature of such optimization problem, and obtaining whose solutions is time- and resource-consuming [11]. Another challenge is the potential impact of system's performance. After consolidation, some VMs may not receive the required resources due to workload dynamic change. As a result, they need be migrated again and again. Hence, VM consolidation may degrade system's performance.

Most of the current research studies about migrating VMs are based on resource utilization since there is a relationship between the total energy consumption by a server and its resource utilization [12]. Whenever resource utilization of a server such as CPU utilization is lower than lower threshold, all VMs hosted on this server are migrated to other server and this unused server is switched to a low-power state. However, it tends to aggressive-migrate for some types of applications such as large capacity applications such as speech recognition, image processing, and decision support systems.

In this study, our research hypothesis is that an aggressive consolidation approach of VM can result in migration thrashing. Our proposal extends current VM consolidation solutions by including constraints that VMs with high capacity are not migrated. The proposed approach not only minimizes the number of servers used but also reduces migration thrashing as far as possible.

The remainder of the paper is organized as follows. Section 2 surveys the related work on VM consolidation in cloud data centers. Section 3 specifies the problem statement. Section 4 discusses the proposed approach. Section 5 presents performance metrics and evaluates the proposed approach. Finally, the conclusion and future work are discussed in Section 6.

2. Related Work

VM consolidation is a key technique in a cloud resource management system to increase the energy efficiency of cloud data centers [13] because the change of user demand and quitting or addition of PMs are continuous events happening in cloud data centers. There is a wide area of research in the VM consolidation field in cloud computing. Most of the studies develop heuristic and metaheuristic approaches and distributed and self-organized approaches. This section presents studies from some of these groups.

The authors in [14] studied the round-robin algorithm to consolidate VMs. They proposed a new strategy for VM consolidation that is called dynamic round-robin (DRR), which is the extension of the round-robin method, and it aims to reduce the number of active servers using two rules. The first rule is that if a VM has finished and there are still other VMs hosting on the same server, this server will not accept new VMs. The second rule is that if a server is in the state satisfying the first rule sufficiently for a long period of time, the server will be forced to migrate the rest of its VMs to other servers, and be shut down after migration finishes.

The authors in [15, 16] proposed a combination of two heuristics for VM consolidation. The first heuristic, called minimization of migrations (MM), selects the VMs that should be migrated from a given server. It is based on the idea of setting upper and lower utilization thresholds for servers. If the utilization of a server drops below the lower threshold, all VMs residing on that server should be removed in order to switch the server to the sleep mode to eliminate the idle power consumption. If the utilization of the server exceeds the upper threshold, some VMs on the server should be removed to reduce the utilization in order to avoid possible SLA violations. The MM heuristic selects the minimum number of VMs to be migrated from a server to lower the CPU utilization below the upper threshold. The other heuristic, called MBFD (modified best-fit decreasing), addresses the mapping of VMs to servers. This can be used for the following purposes: (i) to accommodate new VMs for user requests and (ii) to find a new placement of the VMs selected for migration from the overloaded and underloaded servers. Beloglazov et al. illustrated that MM and MBFD perform well in practice and outperform other competing heuristics [15, 16].

There are some works that treat VM consolidation as an instance of the bin-packing problem to minimize the number of running servers for saving energy. The authors in [17] illustrated the problem of VM consolidation as a bin-packing problem and applied the genetic algorithm and a workload forecasting technique to minimize the energy consumption. The authors in [18] modeled the VM consolidation as multidimensional bin-packing problem and proposed the VM consolidation algorithm based on ant colony optimization (ACO), and the ACO-based approach achieved better energy-performance tradeoff and can be implemented in a distributed environment. In the proposed algorithm, each ant uses the probabilistic decision rule to choose a particular item to pack in its current bin. This rule is based on the pheromone concentration on the item-bin pairs and heuristic information which favors ants choosing the most promising items. The authors in [19] modeled the consolidation problem as a modified multidimensional bin-packing problem. They studied the interrelationships among energy consumption, resource utilization, and performance of consolidated workloads. The authors in [20] modeled VM consolidation as a multidimensional bin-packing problem and proposed a consolidation algorithm—Sercon, which not only minimizes the overall number of used servers but also minimizes the number of migrations. By introducing either of two user-defined parameters, they targeted suboptimal solution to two objective functions. The authors in [21] proposed energy-efficient consolidation policy with a data-intensive energy model based on artificial bee colony (ABC), named as DataABC, and DataABC gets a fast and global optimized decision of VM consolidation.

There are also some works that treat VM consolidation as a mathematics optimization problem. The authors in [22] proposed an improved genetic algorithm for VM consolidation, which achieves better energy-performance tradeoff. The authors in [23] formulated VM consolidation as a multiobjective combinational optimization problem and applied a highly adaptive online optimization metaheuristic called ant colony optimization (ACO) to find a near-optimal solution. The proposed approach uses artificial ants to consolidate VMs into a smaller number of active servers according to current load. The authors in [24] introduced the modified particle swarm optimization (MPSO) method into VM consolidation to avoid falling into local optima. Initially, the VM is assigned on the first physical node meeting the resource requirement by the first-fit algorithm, and N distribution plans are obtained in this step. Each distribution plan is sorted in descending order, and N particles are generated. By using a fitness function, the positions of particles are updated until the personal best of each particle and the global best of all particles are obtained. This method reduces the number of active servers and the amount of VM migrations. The authors in [25] proposed the modified the shuffled frog-leaping algorithm (MSFLA) to solve the dynamic allocation problem of VMs. This algorithm is based on individual meme evolution, which is regarded as a meme carrier in the ethnic group, and global search for information exchange in the entire meme population. Initially, the SFLA randomly generates a frog population, then fitness value of each frog position is calculated, and the frogs are allocated into m groups. A local search is performed in each group, and the last frog with the lowest fitness value in each group is updated until the specified iterations are reached. After that, the entire frog is remixed and sorted, and the local depth search is performed again. The search ends when the termination condition is satisfied. The authors in [26] considered energy of processor, disk, and migration, and performance degradation when VMs are consolidated, formulated VM consolidation as a multiobjective optimization problem. They used simulated annealing (SA) to solve the mentioned optimization problem.

The authors in [27] proposed a novel distributed VM consolidation mechanism. They used game theory to model that each VM is the player which can choose any server as a strategy. The player pays a cost (such as power consumption) depending on its strategy and strategies of the other players. Since all the players prefer to pay less, they conduce to servers with the least remainder of capacity. After that, the underloaded server is shut down in order to reduce energy consumption.

There are some works that regard VM consolidation as a matchmaking problem. The authors in [28] expressed IT managers' preferences and metrics, applied matchmaking to locate entities (i.e., VMs and servers) which best match each other's preferences.

The authors in [29] presented a linear programming formulation and added constraints to control VM migration on VM consolidation process. They considered migration control constraints with CPU and memory to avoid unnecessary migrations in order to minimize performance penalty.

All the aforementioned works only considered VM instant resource usage when VMs are consolidated. This paper considers the fact that VMs with high capacity are not suitable as a migration candidate since it can cause migration thrashing which will make these virtual machines suffer from performance loss. Besides, by keeping those VMs on their original physical servers to reduce migration thrashing, the total number of migrations is minimized.

3. Problem Statement

In cloud data centers, the problem of VMs distributed on a pool of servers can be mapped to the bin-packing problem, and the goal is that VMs with different sizes must be packed into a finite number of servers so that minimum servers are used. This problem is NP-hard, and a number of heuristic algorithms are provided to give suboptimal solutions, such as first-fit, next-fit, best-fit, and best-fit decreasing.

Dynamic consolidation (DC) can perform the required configuration changes according to the current load resource demands by utilizing VM migration which enables the transference of VMs among physical servers with negligible downtime and maintaining sessions active. However, DC usually leads to a degradation of VM's performance since some amount of CPU time is spent processing on the migrating server, and a portion of link bandwidth is utilized between the migrating and migrated servers, etc. Especially, DC can result in that the same VM migrates again and again across servers, and we call it migration thrashing (MT). Figure 1 illustrates an example to demonstrate MT of DC. For the convenience of presentation, we suppose that physical machines (PMs) have the same amount of resources, and hence, resource demands by each VM can be denoted in portion. To determine which VMs should be migrated, we employ minimization of migrations (MM) policy which selects a minimum number of VMs to migrate to lower CPU utilization below the upper utilization threshold if the upper threshold is violated. Note that each VM can change its demand at any time since workload can scale down or scale up in a matter of minutes or even seconds. In our case, upper utilization threshold of PM is 70%. Figure 1(a) shows eight VMs distributed on three PMs in the initial state, and hence, PM_1 is overloaded (utilization is 90%). Figure 1(b) shows that VM_1 has to be migrated from PM_1 to PM_2 according to the MM policy. Figure 1(c) shows that both VM_5 and VM_6 in PM_2 change their demands to 10% at a time step and PM_2 is overloaded. Figure 1(d) shows that VM_1 is migrated to PM_3 because both PM_1 and PM_2 have not enough room to hold it. Figure 1(e) shows that all VMs in PM_1 change their resource demands to 3% at a time step, both VM_7 and VM_8 in PM_3 change their demands to 10% at the same time step, and PM_3 is overloaded. Figure 1(f) shows that VM_1 is migrated from PM_3 back to PM_1.

It is obvious that the VMs with high capacity usually are treated as the migration objects by existing selection policies (i.e., MM policy) due to enabling to reduce the number of migrations. In nature, candidate target VMs should be selected elaborately to alleviate MT. However, traditional VM

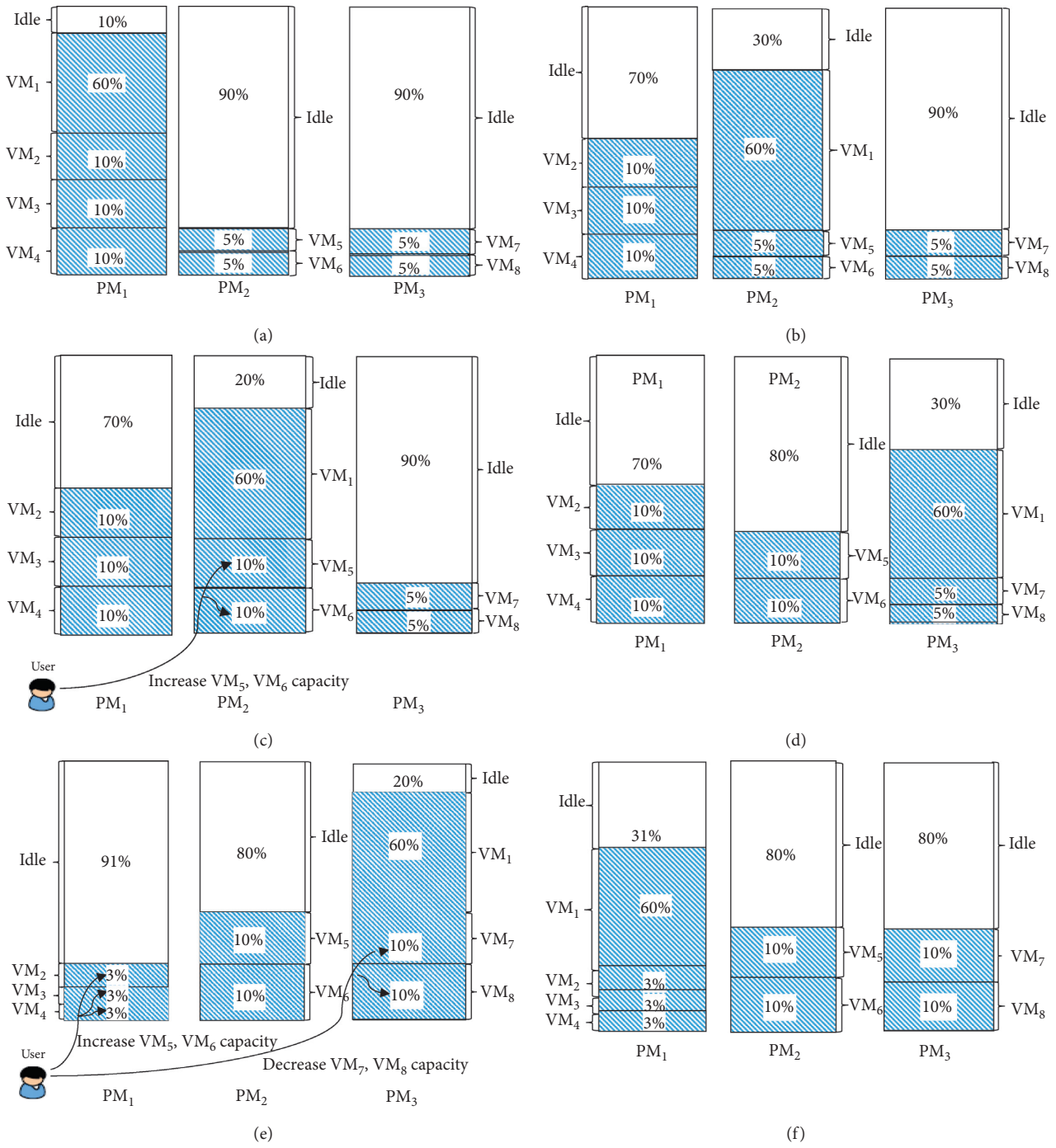

Figure 1: Migration thrashing of dynamic consolidation. (a) Initial state. (b) VM_1 is migrated to PM_2. (c) PM_2 is overloaded because VM_5 and VM_6 scale up. (d) VM_1 is migrated to PM_3. (e) VM_2, VM_3, and VM_4 scale down and PM_3 is overloaded because VM_7 and VM_8 scale up. (f) VM_1 is migrated back to PM_1.

selection policies generate a high MT of VMs because VMs are consolidated according to their instant resource usage, instead of considering their overall and long-term utilization when making a decision.

In the following sections, we present an approach to minimize MT of VMs in DC and we define the resulting problem as dynamic consolidation with minimization of MT (DCMMT).

4. Dynamic Consolidation with Minimization of Migration Thrashing

4.1. Problem Formulation. The DC problem is modeled as a stochastic bin-packing problem (SBPP) usually. Servers and VMs are modeled as bins and items with each type resource (CPU, memory, disk space, network bandwidth, etc.). Volumes of bins representing capacities of servers are

usually definitized, since they can be derived by hardware configuration of servers. Volumes of items representing resource demands of VMs are random variables, since they are usually time-varying and rely on workloads running inside VMs. The DC problem aims at minimizing the number of required physical servers, with the constraint that each VM is mapped to a single physical server and the aggregated resource consumption of VMs on a physical server does not overload server capacity.

The formal definition of the problem is given as follows:

P: set of physical servers

V: set of VMs

R: set of resources (CPU, memory, disk space, and network bandwidth)

C_p^r: nonnegative capacity for physical server $p \in P$ of resource $r \in \mathrm{R}$

D_v^r: nonnegative random demand of VM $v \in V$ for resource $r \in \mathrm{R}$

x_{pv}: VM-to-PM assignment Boolean variable which is equal to 1 if $v \in V$ is mapped to server $p \in P$, 0 otherwise

y_p: server selection Boolean variable which is equal to 1 if server $p \in P$ is used, 0 otherwise

The objective is to select the required physical servers in order to minimize the number of used servers:

$$\min = \sum_{p \in P} y_p, \tag{1}$$

Subject to

$$\sum_{p \in P} x_{pv} = 1, \quad \forall v \in V, \tag{2}$$

$$\sum_{v \in V} D_v^r x_{pv} \le C_p^r y_p, \quad \forall p \in P, \forall r \in R, \tag{3}$$

$$x_{pv} \in \{0, 1\}, \quad \forall p \in P, \forall v \in V, \tag{4}$$

$$y_p \in \{0, 1\}, \quad \forall p \in P. \tag{5}$$

Objective function (1) minimizes the number of used servers. Constraint (2) ensures that each VM is loaded into one server at most. Constraint (3) ensures that the VM resource demands do not exceed server capacity. Constraints (4) and (5) are integrality constraints. In each consolidation step, VMs may be migrated to different servers, which results in a diverse mapping of VMs to servers and causes servers that are not running VMs (idle servers) to be shut down or in a low-power mode to eliminate idle power. The number of required servers can change after each consolidation step.

In order to as far as possible avoid penalizing performance of the VM involved in MT, we make some revisions to the original problem. A parameter and a constraint are added to the original problem. In DCMMT, we keep VMs likely to suffer from MT in the same servers. The parameter C_v included in DCMMT is defined as capacity coefficient in Table 1, which is a safety parameter of the method that defines VMs with how much capacity are kept in the same servers in current consolidation step. In other words, C_v decides how aggressively the system migrates VMs. In DC, the target VM for migration is selected only aiming at lowering CPU utilization below upper threshold, but in DCMMT, the resource demand of target VM must also satisfy inequality (6). For example, a VM is selected by a selection policy, if its resource demand exceeds the product of C_v and utilization upper threshold and it is prohibited to migrate. When C_v is set to 0, there was no VM migration and then no MT, but it is not in line with the idea of VM consolidation and no practical significance. When C_v is set to 1, DCMMT turns back to DC. Optimal value of parameter C_v is obtained by experiment.

Table 1: New parameters included in DCMMT.

Parameters	
$C_v \in (0, 1)$	Capacity coefficient, more than 0 and less than 1

$$D_v^r \le C_v * \text{UpperThreshold}, \quad \forall v \in V, \forall r \in R. \tag{6}$$

Upper threshold and lower threshold are utilization upper threshold and lower threshold. The constraint included in DCMMT is presented in the following inequality:

4.2. Algorithm Design. The basic DC problem is divided into four parts: (1) determining when a server is considered as being overloaded; (2) determining when a server is considered as being underloaded; (3) selection of VMs which should be migrated from an overloaded server; (4) finding a new placement for VMs to be migrated from the overloaded and underloaded servers. This paper focuses on the third and the fourth phases. In DCMMT, the heuristics are revised through modifying the third phase in order to minimizing MT.

The bin-packing problem can be solved using existing solvers. Such approaches include integer linear programming, pseudo-Boolean optimization, mixed integer nonlinear programming, and binary integer programming. Their solution is guaranteed to be the best. However, all of these methods have scalability issues because runtime is incredibly long for moderate-sized data centers. Therefore, these methods are not available in practice. Although heuristics do not ensure an optimal solution, the required time to obtain a feasible solution is more acceptable than above approaches, especially while handling large-scale scenarios. Best-fit decreasing (BFD) is shown to use no more than $11/9\mathrm{OPT} + 1$ bins (where OPT is the number of bins provided by optimal solution). So, we do a little modification in existing heuristics to achieve that VMs with high capacity are not migrated in each consolidation step. The pseudocode of the algorithm that selects the VMs which should be migrated is presented in Algorithm 1. Firstly, the algorithm initializes the migration sign of each VM in the VM list. Secondly, the algorithm sorts the list of VMs in decreasing order of the CPU utilization. Then, it traverses the list of VMs and finds a VM to migrate, and at this time, three conditions are considered: first, the VMs whose utilization is

higher than the difference between server utilization and upper threshold are the candidates; second, if a VM is selected to migrate, the difference between upper threshold and server utilization after migration is the minimum across the candidates. If there is no such a VM, the algorithm selects the VM with highest utilization and continues to find more VMs to migrate. Third, according to equation (6), only the VMs which satisfy migration constraint are selected to migrate. If a VM can be migrated, we add it to the migration list of VMs otherwise remove it from the VM list of the server to continue the next iteration to find a new VM. The algorithm stops when new utilization of the server drops below upper utilization threshold. For the underloaded server, we migrate all VMs on it and then shut down it. The complexity of the algorithm is proportional to the product of the number of overloaded servers and the number of VMs allocated to each overloaded server.

The pseudocode for the VM placement algorithm is presented in Algorithm 2. The algorithm sorts all VMs in decreasing order of their current CPU utilization and sorts all servers in ascending order of their current CPU utilization. And each VM is assigned to the first server with sufficient resources taking into account upper threshold. If there is no such a server, we start a new server for VM, allocate VM on it, sort a new server list again, and proceed to a new iteration. The complexity of the algorithm is proportional to the product of the number of VMs which need to be migrated and the number of all servers.

5. Evaluation

5.1. Performance Metrics. In order to compare the efficiency of the algorithms, we use several metrics to evaluate performance.

5.1.1. Thrashing Index (TI) Metric. This metric is proposed by us and defined for measuring MT of a VM. It is the ratio of the number of migrations to the number of consolidation steps during whole execution. The TI varies from 0 to 1, and the larger the value, the more severs the MT of VM. This metric is calculated as shown in the following equation:

$$\mathrm{TI} = \frac{N_{\mathrm{m}}}{N_{\mathrm{c}}}, \tag{7}$$

where N_{m} is number of migrations of VMs and N_{c} is number of consolidation steps during whole execution.

5.1.2. SLA Violation Metric. In cloud computing, a service-level agreement (SLA) is an agreement between a service provider and a consumer. To abide by the SLA, the service provider must guarantee quality of service (QoS) closely through such performance metrics as computing capacity, response time, throughput, timeout, etc. According to [14], SLA violation metrics can be classified into two groups: (1) the percentage of time, during which active servers have experienced CPU utilization of 100%, SLA violation time per active host (SLATAH), and (2) the overall performance degradation by VMs due to migrations, performance degradation due to migrations (PDMs). These two metrics are calculated as follows:

$$\begin{aligned} \mathrm{SLATAH} &= \frac{1}{N}\sum_{i=1}^{N}\frac{T_{si}}{T_{ai}}, \\ \mathrm{PDM} &= \frac{1}{M}\sum_{j=1}^{M}\frac{C_{dj}}{C_{rj}}, \end{aligned} \tag{8}$$

where N is the number of servers; T_{si} is the total time during which the server i has experienced the utilization of 100%; T_{ai} is the total time during which the server i is in the active mode; and M is the number of VMs. C_{dj} is the estimate of the performance degradation of the VM_j caused by migration; C_{rj} is the total CPU capacity requested by the VM_j during its lifetime. Both the SLATAH and PDM metrics are independent of each other and are of equal importance to the level of SLA violations. Consequently, (9) defines a general metric for calculating the total value of SLA violations that embodies performance degradation due to both server overloading and VM migration:

$$\mathrm{SLAV} = \mathrm{SLATAH} \cdot \mathrm{PDM}. \tag{9}$$

Another metric we used for calculating the SLA violation is average SLA violation, which is mean of the difference between the requested million instructions per second (MIPS) by a VM and the allocated MIPS divided by the requested MIPS as shown in (10). Here, N is the number of VMs:

$$\text{average SLA violation} = \frac{\sum_{i=1}^{N}(\text{requestedMIPS} - \text{allocatedMIPS}))/\text{requestedMIPS}}{N}. \tag{10}$$

5.1.3. Average Number of VM Migrations. This metric is the average number of VM migrations in all consolidation steps with respective standard deviations during a data center processes all workloads.

5.2. Experiment Setup. Since the target system is an infrastructure-as-a-service (IaaS) and a generic cloud computing environment that offers a view of limitless computing resources to users, it is vital to evaluate the proposed consolidation algorithms on a large-scale virtualized data center infrastructure. However, implementing and evaluating the proposed algorithms on real infrastructure is very expensive and time-consuming. Furthermore, it is hard to executing repeatable large-scale experiments to analyze and compare the results of proposed algorithms since the result cannot be

```
(1) Input: serversList Output: migrationList
(2) FOR s IN serverList
(3)    vmList ⟵ s.getVmList()
(4)    FOR v IN vmList
(5)       v.canMigrated = 1
(6)    descVms ⟵ sortDecreasing(vmList)
(7)    sUtil ⟵ s.getUtil()
(8)    bestDiff ⟵ MAX
(9)    WHILE sUtil > UpperThreshold
(10)      FOR vm IN descVms
(11)         IF vm.getUtil() > sUtil–UpperThreshold
(12)            t ⟵ vm.getUtil()–(sUtil–UpperThreshold)
(13)            IF t < bestDiff
(14)               bestDiff ⟵ t
(15)               bestFitVm ⟵ vm
(16)            END
(17)         ELSE IF bestDiff = = MAX
(18)                  bestFitVm ⟵ vm
(19)               END
(20)               break
(21)         END
          END
(22)      reset bestFitVm.canMigrated according to equation (6)
(23)      IF bestFitVm.canMigrated
(24)         sUtil ⟵ sUtil–bestFitVm.getUtil()
(25)         migrationList.add(bestFitVm)
(26)         descVms.remove(bestFitVm)
(27)      ELSE
(28)         //mark bestFitVm being not migrated, and remove it from descVms
(29)         bestFitVm.canMigrated = 0
(30)         descVms.remove(bestFitVm)
(31)      END
(32)   END
(33)   IF sUtil < LowerThreshold
(34)      migrationList.add(s.getVmList())
(35)      descVms.remove(s.getVmList())
(36)      serversList.remove(s)
(37)   END
(38) END
(39) return migrationList
```

Algorithm 1: Selection of VMs which should be migrated.

reproduced on real infrastructure. Therefore, a simulation has been chosen as a way to evaluate the proposed algorithms. The CloudSim Toolkit 4.0 [30] has been chosen as a simulation platform as it is a modern simulation framework aimed at cloud computing environments. It supports the modeling of virtualized data centers, enables on-demand resource provisioning, provides modeling of dynamic workload, and has been extended to account for heterogeneous cloud power-aware simulations, server processor SLEEP state transitions, different VM placement and thermal control heuristics, etc. [31].

In our infrastructure setup, we have simulated a cloud computing infrastructure comprising a data center with 800 heterogeneous physical machines including 200 HP ProLiant ML110 G4, 200 HP ProLiant ML110 G5, 200 IBM System x3630 M3, and 200 IBM System x3755 M3. The characteristics of these machines are depicted in Table 2. Each server was modeled to have 1 GB/s network bandwidth. These characteristics correspond to five Amazon EC2 VM instance types which are depicted in Table 3. Since using real workload for simulation experiments is significant, we have used the real-world workloads provided as a part of the CoMon project which is a monitoring infrastructure for PlanetLab. Workload data of CPU utilization in 5 min intervals by more than a thousand VMs from servers located at more than 500 places around the world have been used (Table 4), and the value of workload data confirms that average CPU utilization is far below 50%. Three different workload data collected from three different days have been applied. During the simulation, each VM is randomly assigned a workload trace from a certain day. Initially, VMs are assigned resources according to resource demands defined by each VM instance type. During lifetime of VM, its CPU utilization varies according to workload trace data

```
(1) Input: serversList, vmsToMigrate Output: migrationMap, which is a type of Map<Server, Vm>
(2) ascServers ← sortAsending(serverList)
(3) descVms ← sortDecreasing(vmsToMigrate)
(4) FOR vm IN descVms
(5)     allocatedServer ← NULL
(6)     FOR server IN ascServer
(7)       sUtil ← s.getUtil()
(8)       IF sUtil + vm.getUtil() < UpperThreshold
(9)         allocatedServer ← server
(10)        migrationMap.put(allocatedServer, vm)
(11)        descVms.remove(vm)
(12)        break
(13)      END
(14)    END
(15)    IF allocatedServer = = NULL
(16)      server ← create a new server for vm
(17)      allocatedServer ← server
(18)      migrationMap.put(allocatedServer, vm)
(19)      descVms.remove(vm)
(20)      newServerList ← ascServers
(21)      newServerList.add(server)
(22)      ascServers ← sortAsending(newServerList)
(23)    END
(24) END
(25) return migrationMap
```

Algorithm 2: VM placement.

Table 2: Configuration of servers.

Server	CPU model	Cores	Frequency (MHz)	RAM (GB)
HP ProLiant ML110 G4	Intel Xeon 3040	2	1860	4
HP ProLiant ML110 G5	Intel Xeon 3075	2	2660	4
IBM System x3630 M3	Intel Xeon X5675	12	3067	4
IBM System x3755 M3	AMD Opteron 6276	32	2300	32

Table 3: VM types (five EC2 VM types).

VM type	CPU (MIPS)	RAM (GB)
High-Memory Extra Large Instance	3000	8.5
High-CPU Medium Instance	2500	0.87
Extra Large Instance	2000	3.75
Small Instance	1000	1.74
Micro Instance	500	0.613

Table 4: Workload data characteristics (CPU utilization rate).

Data	Num. of VMs	Mean (%)	St. dev (%)	Quartile1 (%)	Median (%)	Quartile3 (%)
Workload1	1052	12.31	17.09	2	6	15
Workload2	898	11.44	16.83	2	5	13
Workload3	1516	9.26	12.78	2	5	12

without exceeding its initial value, and consequently, DC is required.

As mentioned before, DC is divided into four subproblems. For all the experiments, in order to address the subproblems determining whether or not the server is overloaded/underloaded, the static threshold (ST) VM allocation policy has been chosen. Two remaining subproblems are solved by using the proposed algorithm.

Baseline for comparison: in order to evaluate benefits of our approach, we have compared our proposed DCMMT strategy with four ST consolidation strategies which do not deal with MT and comprise two parts (selection of VMs and

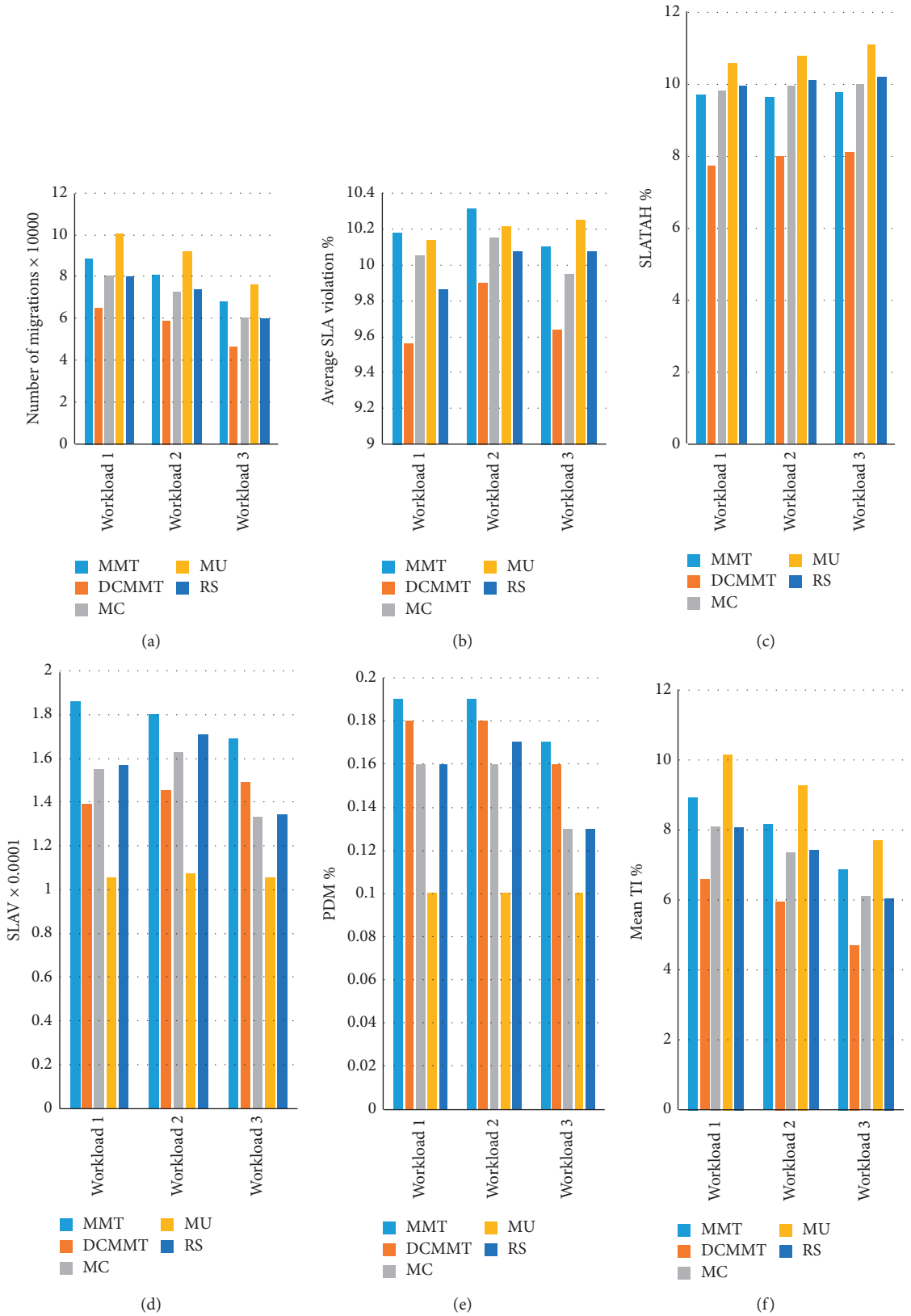

Figure 2: Comparison of the DC, DCMMT, MC, MU, and RS. (a) Number of migrations, (b) average SLA violation, (c) SLATAH, (d) SLAV, (e) PDM, and (f) mean thrashing index.

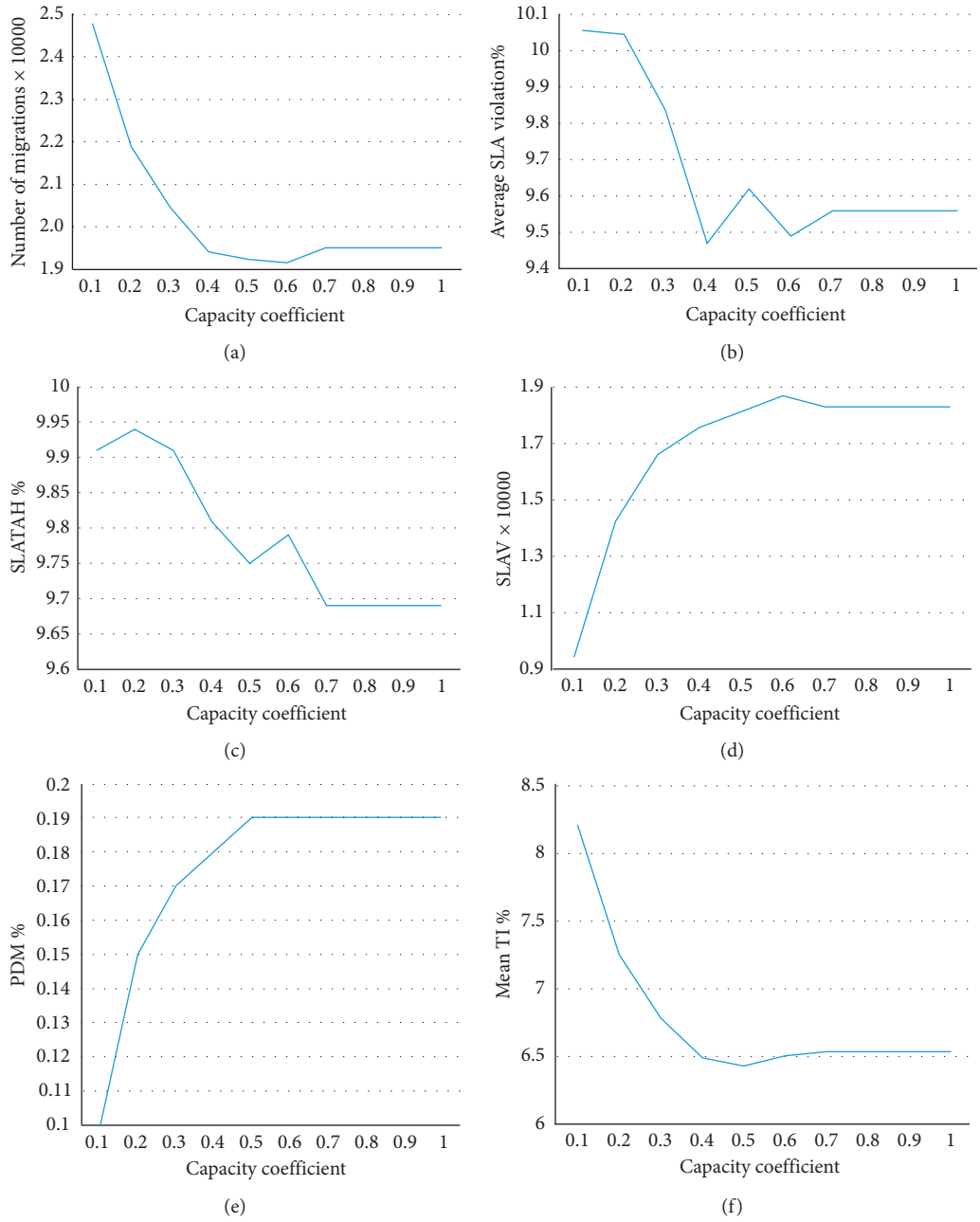

Figure 3: Performance metrics with the workload1 data trace under different capacity coefficients.

placement of VMs) similar to our approach. As for the placement of VMs, both our approach and the approaches for comparison use Algorithm 2. The selection strategies implemented by the approaches for comparison were as follows:

Minimum migration time (MMT) strategy. This strategy migrates a VM that requires minimum migration time relatively to other VMs on the server.

Maximum correlation (MC) strategy. This strategy selects a VM to be migrated that has highest correlation of CPU utilization with other VMs.

Minimum utilization (MU) strategy. This strategy selects a VM to be migrated that has minimum CPU utilization.

The random selection (RS) strategy. This strategy selects a VM to be migrated according to a uniform distribution.

For simplicity, we identify diverse consolidation strategy by the name of selection strategy.

5.3. Simulation Results. Using the workload data described in Section 5.1, we have simulated the DCMMT approach and four static threshold consolidation approaches which do not deal with MT. Capacity coefficient is set to 0.7. The results depicted in Figure 2 show that DCMMT significantly outperforms other approaches. For example, Experiment-1 shows that the proposed algorithm reduces the number of

migrations by up to 26.93%, average SLA violation up to 6.09%, SLATAH up to 20.12%, SLAV up to 25.38%, and PDM up to 5.23%, and reduces the mean thrashing index by up to 26.13% than MMT while using workload1, etc.

Experiment-2 shows that the proposed algorithm reduces the number of migrations by up to 29.45%, SLA violation up to 3.98%, SLATAH up to 16.99%, SLAV up to 19.42%, and PDM up to 5.23%, and reduces the mean thrashing index by up to 26.95% than MMT while using workload2, etc.

Experiment-3 shows that the proposed algorithm reduces the number of migrations by up to 31.59%, SLA violation up to 4.55%, SLATAH up to 17.16%, SLAV up to 11.83%, and PDM up to 5.88%, and reduces the mean thrashing index by up to 31.47% than MMT while using workload3, etc.

One of drawbacks of DC (e.g., MMT, MC, MU, and RS) is bringing about MT when the same VM migrates again and again across servers. Our proposed algorithm benefits from more restrictive decision-making which limits aggressive consolidation. Generally, the proposed algorithm tries to identify the VMs that are more suited to be migrated. Due to better selections of VM to be migrated, the number of migrations in our algorithm is reduced because controlling migration of the VMs from overloaded servers eliminates MT which can significantly increase the number of migrations. It is depicted in Figure 2(a) with three different workloads, and it directly affects PDM as shown in Figure 2(e), especially to the VMs which can be migrated again and again across servers, and these VMs avoid performance penalty brought by MT. Figure 2(b) depicts that in our algorithm, the metric SLATAH has been decreased. In fact, since our algorithm prevents the VMs being migrated again and again, the situation that application running inside VMs cannot be responded or response time is too long during migration has been alleviated compared with DC policy. Due to better selections of VM to be migrated, the time of experiencing 100% utilization in our algorithm is reduced by decreasing the number of migrations of the VMs with high capacity, and it is depicted in Figure 2(c). As a result, our algorithm decreases the SLAV which is a combined metric (i.e., the product of the PDM and SLATAH) as shown in Figure 2(d).

In comparison with the DC policy, our proposed algorithm considers MT of a VM to make better decision about being migrated or not. As a result, the thrashing index is reduced by controlling migration of the VMs from overloaded servers as shown in Figure 2(f).

5.4. Sensitivity Analysis. Here, we study effect of changing the capacity coefficient in DCMMT to determine the reasonable value of capacity coefficient that gives preferable performance by conducting the simulations with a single workload. We use different capacity coefficients (ranging from 0.1 to 1.0). The demands of VMs are drawn from a workload1 data trace. Figure 3 shows performance metrics. In general, as capacity coefficient increases, the number of migrations decreases as shown in Figure 3(a), which is explained by the smaller the capacity coefficient, the more the VMs with low capacity to be migrated. However, when capacity coefficient is greater than 0.7, the number of migrations does not decrease anymore, since VMs with capacity higher than threshold are less and less because most of VMs are small in workload1 (i.e., mean value of CPU utilization of workload1 is 12.31%). Similar explanations may pertain to Figures 3(b), 3(c), and 3(f). In Figure 3(e), as capacity coefficient increases, PDM increases. Because the more the VMs with high capacity to be migrated, the more the possibility to cause a new server experiencing 100% utilization. However, PDM does not decrease when capacity coefficient is greater than 0.5, since VMs with the capacity higher than threshold are less and less. Figure 3(d) is a combined reflect of Figures 3(c) and 3(e). Consequently, the optimal capacity coefficient was obtained using a value ranging from 0.7 to 1. Of course, the optimal capacity coefficient is affected by workload data characteristics.

6. Conclusions

In order to minimize MT in cloud data centers, we proposed an approach for DC of VMs with minimization of MT. We also provided analysis of our approach using linear programming formulation and heuristics. To the best our knowledge, this is the first work on VM migration thrashing. The main difference from existing solutions on DC is that our approach identifies VMs with appropriate capacity. This is achieved by including a constraint that VMs with high capacity are not migrated when the server is overloaded. We evaluated the proposed algorithms through simulations on a large-scale experiment setup using a workload data trace from more than a thousand PlanetLab VMs.

The experiment results showed that the proposed algorithm outperforms DC algorithms in regard to the SLAV metric due to a reduced level of SLA violations and the TI metric due to a reduced level of the number of VM migrations. The results obtained from these experiments were promising as data centers can easily benefit from our mechanism since little modifications are required to be integrated into a current resource management system.

For our future's work, for the sake of evaluating the proposed algorithm in a real cloud infrastructure, we would like to implement it by extending a real-world cloud platform, such as OpenStack. Other directions for future work are the investigation of workload statistical properties, including cumulative distribution function (CDF), coefficient of variation (COV), correlation among workloads belonging to nodes, and periodicity or stability, and investigation of workload models, e.g., models based on Markov chains, as well as developing algorithms that will exploit these findings.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

The funders did not influence the study design, the collection, analysis, and interpretation of data, the writing of the

manuscript, and the decision to submit the manuscript for publication.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was funded by the Science and Technology Department of Shaanxi Province, grant number: 2019GY-020.

References

[1] A. Mohan and Shine, "Survey on live vm migration techniques," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 2, pp. 60–74, 2013.

[2] K. Jonathan, *Growth in Datacenter Electricity Use 2005 to 2010*, Analytics Press, Oakland, CA, USA, 2011, http://www.analyticspress.com/datacenters.html.

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

[4] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 13–23, 2007.

[5] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2008.

[6] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[7] A. Abdelsamea, A. A. El-Moursy, E. E. Hemayed, and H. Eldeeb, "Virtual machine consolidation enhancement using hybrid regression algorithms," *Egyptian Informatics Journal*, vol. 18, no. 3, pp. 161–170, 2017.

[8] R. Rose, "Survey of system virtualization techniques," *Virtualization*, vol. 3, pp. 1–15, 2004.

[9] M. Noshy, A. Ibrahim, and H. A. Ali, "Optimization of live virtual machine migration in cloud computing: a survey and future directions," *Journal of Network and Computer Applications*, vol. 110, pp. 1–10, 2018.

[10] R. Nathuji and K. Schwan, "VirtualPower: coordinated power management in virtualized enterprise systems," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, pp. 265–278, 2007.

[11] M. H. Malekloo, N. Kara, and M. E. Barachi, "An energy efficient and SLA compliant approach for resource allocation and consolidation in cloud computing environments," *Sustainable Computing: Informatics and Systems*, vol. 17, pp. 9–24, 2018.

[12] G. Dhiman, K. Mihic, and T. Rosing, "A system for online power prediction in virtualized environments using Gaussian mixture models," in *Proceedings of the 47th Design Automation Conference on—DAC '10*, pp. 807–812, Anaheim, CA, USA, June 2010.

[13] M. A. Khan, A. Paplinski, A. M. Khan, M. Murshed, and R. Buyya, "Dynamic virtual machine consolidation algorithms for energy-efficient cloud resource management: a review," *Sustainable Cloud and Energy Services*, Springer, Cham, Switzerland, pp. 135–165, 2018.

[14] C.-C. Lin, P. Liu, and J.-J. Wu, "Energy-aware virtual machine dynamic provision and scheduling for cloud computing," in *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing*, Washington, DC, USA, July 2011.

[15] A. Beloglazov and R. Buyya, "Energy efficient allocation of virtual machines in cloud data centers," in *Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, Melbourne, Australia, May 2010.

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

[17] H. Hlavacs and Treutner, "Genetic algorithms for energy efficient virtualized data centers," in *Proceedings of the 2012 8th International Conference on Network and Service Management (CNSM) and 2012 Workshop on Systems Virtualiztion Management (SVM)*, pp. 422–429, Las Vegas, NV, USA, October 2012.

[18] E. Feller, L. Rilling, and C. Morin, "Energy-aware ant colony based workload placement in clouds," in *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing GRID '11*, pp. 26–33, IEEE Computer Society, Lyon, France, September 2011.

[19] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, San Diego, CA, USA, February 2008.

[20] A. Murtazaev and S. Oh, "Sercon: server consolidation algorithm using live migration of virtual machines for green computing," *IETE Technical Review*, vol. 28, no. 3, p. 212, 2011.

[21] J. Jiang, Y. Feng, J. Zhao, and K. Li, "DataABC: a fast ABC based energy-efficient live VM consolidation policy with data-intensive energy evaluation model," *Future Generation Computer Systems*, vol. 74, pp. 132–141, 2016.

[22] J. Xu and J. A. B. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Proceedings of the 2010 IEEE/ACM International Conference on Green Computing and Communications and International Conference on Cyber, Physical and Social Computing*, pp. 179–188, Hangzhou, China, December 2010.

[23] F. Farahnakian, A. Ashraf, T. Pahikkala et al., "Using ant colony system to consolidate VMS for green cloud computing," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 187–198, 2015.

[24] H. Li, G. Zhu, C. Cui, H. Tang, Y. Dou, and C. He, "Energy-efficient migration and consolidation algorithm of virtual machines in data centers for cloud computing," *Computing*, vol. 98, no. 3, pp. 303–317, 2016.

[25] J.-p. Luo, X. Li, and M.-r. Chen, "Hybrid shuffled frog leaping algorithm for energy-efficient dynamic consolidation of virtual machines in cloud data centers," *Expert Systems with Applications*, vol. 41, no. 13, pp. 5804–5816, 2014.

[26] M. Sharifi, H. Salimi, and M. Najafzadeh, "Power-efficient distributed scheduling of virtual machines using workload-aware consolidation techniques," *The Journal of Supercomputing*, vol. 61, no. 1, pp. 46–66, 2012.

[27] H. Khani, L. Amin, N. Yazdani, and S. Mohammadi, "Distributed consolidation of virtual machines for power efficiency in heterogeneous cloud data centers," *Computers & Electrical Engineering*, vol. 47, pp. 173–185, 2015.

[28] X. Li, A. Ventresque, J. Murphy, and J. Thorburn, "SOC: satisfaction-oriented virtual machine consolidation in

enterprise data centers," *International Journal of Parallel Programming*, vol. 44, no. 1, pp. 130–150, 2016.

[29] T. C. Ferreto, M. A. S. Netto, R. N. Calheiros, and C. A. F. De Rose, "Server consolidation with migration control for virtualized data centers," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1027–1034, 2011.

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

[31] M. R. Velayudhan Kumar and S. Raghunathan, "Heterogeneity and thermal aware adaptive heuristics for energy efficient consolidation of virtual machines in infrastructure clouds," *Journal of Computer and System Sciences*, vol. 82, no. 2, pp. 191–212, 2016.