Virtual Machine Consolidation for Cloud Data Centers Using Parameter-Based Adaptive Allocation

Abdelkhalik Mosa
The University of Manchester
Manchester M13 9PL, UK
abdelkhalik.mosa@manchester.ac.uk

Rizos Sakellariou
The University of Manchester
Manchester M13 9PL, UK
rizos@manchester.ac.uk

ABSTRACT

Cloud computing enables cloud providers to offer computing infrastructure as a service (IaaS) in the form of virtual machines (VMs). Cloud management platforms automate the allocation of VMs to physical machines (PMs). An adaptive VM allocation policy is required to handle changes in the cloud environment and utilize the PMs efficiently. In the literature, adaptive VM allocation is typically performed using either reservation-based or demand-based allocation. In this work, we have developed a *parameter-based* VM consolidation solution that aims to mitigate the issues with the reservation-based and demand-based solutions. This parameter-based VM consolidation exploits the range between demand-based and reservation-based finding VM to PM allocations that strike a delicate balance according to cloud providers' goals. Experiments conducted using CloudSim show how the proposed parameter-based solution gives a cloud provider the flexibility to manage the trade-off between utilization and other requirements.

CCS CONCEPTS

•Computer systems organization → Cloud computing; Self-organizing autonomic computing; •Applied computing → Data centers; •Theory of computation → Scheduling algorithms; **•Social and professional topics** → Pricing and resource allocation;

KEYWORDS

Cloud data centers, Virtual Machine mapping, Virtual Machine consolidation, efficient data center utilization

ACM Reference format:
Abdelkhalik Mosa and Rizos Sakellariou. 2017. Virtual Machine Consolidation for Cloud Data Centers Using Parameter-Based Adaptive Allocation. In *Proceedings of ECBS '17, Cyprus, Aug 31-Sep 1,2017,* 10 pages.
DOI: 10.1145/3123779.3123807

1 INTRODUCTION

The main idea behind cloud computing is providing computing resources (*e.g.*, infrastructure, platform or software) as a *service* over a *network* or the Internet. Cloud services are supposed to be *autonomic*, *on-demand*, *elastic* and follow a *pay-as-you-go* utility model in case of public clouds. The cloud service models enable the cloud customers to easily scale a service up or down based on the fluctuation of their needs. The inherent elasticity of the cloud lures cloud customers to use cloud services whenever convenient. In the infrastructure as a service (IaaS) paradigm, cloud data centers multiplex multiple virtual machines (VMs) on a single physical machine (PM) with the help of virtualization technology.

Cloud providers try to efficiently manage the mapping of the virtual machines (VMs) onto physical machines (PMs); a survey of the research done in the area of VM mapping has been conducted by Pietri *et al.*[25]. Existing cloud management tools, also called cloud operating systems, such as *OpenStack*[1] and *OpenNebula*[2] with the embedded virtual machine managers (VMMs) such as *KVM*[3], *VMware ESXi*[4], *or Xen*[5] may manage a data center's infrastructures to build private, public, hybrid or community clouds. These cloud management tools have autonomic controllers that manage the VM-to-PM mapping. This autonomic virtual machine management involves both the *initial VM placement* as well as the adaption to changes in the cloud environment through the *dynamic reallocation* (*consolidation*) of the VMs. The initial VM placement does not react to changes in the cloud environment by the live migration of the VMs. Dynamic VM consolidation is crucial for utilizing a data center's resources efficiently by either reacting or pro-acting to changes in the cloud data center. The dynamic consolidation of the VMs could be either *reservation-based* or *demand-based*. In the *reservation-based* placement, the VM placement decision depends on the amount of *reserved resources* defined by the VM size in terms of CPU, memory and bandwidth capacities rather than the *actual demand*. Reservation-based VM placement ensures better availability and durability; however, it may lead to underutilization of the PMs [19].

Initial and reservation-based VM placement solutions do not utilize the data center's PMs efficiently. Cloud data centers are mostly over-provisioned and the PMs are only 10-50% utilized most of the time [1]. Studies showed that the average CPU utilization is less than 50% [2]. PM underutilization wastes energy as completely idle PMs can consume around 70% of their peak power [11] due to the narrow dynamic power range of the PMs. The Uptime Institute Network [16] listed power efficiency and hardware consolidation for reducing costs and utilizing data centers efficiently as two of the top-ten data center management priorities for 2013. Switching PMs off or to lower power modes could save energy [10]. The main

ECBS '17, Cyprus

DOI: 10.1145/3123779.3123807

[1] https://www.openstack.org/
[2] https://opennebula.org/
[3] https://www.linux-kvm.org/
[4] http://www.vmware.com/products/esxi-and-esx.html
[5] https://www.xenproject.org/

causes of energy waste are the underutilization of the PMs [24]. Accomplishing efficiency with big cloud providers does not solve the problem as the major cloud providers only account for about 5% while the majority of providers concerns small and medium size governmental and corporate data centers [24].

Demand-based placement assigns VMs to PMs based on the *actual (real-time) demand* rather than the *reserved resources*. The demand-based placement approach requires knowing the resource demand that can be identified using *reactive*, *predictive* or *hybrid* methods [15]. The dynamic VM placement strategy migrates VMs from underutilized PMs and then switches those PMs into one of the power saving modes. Thus, through VM consolidation, the number of running PMs is minimized. Reservation-based placement does not lead to PM overload, but the PMs are underutilized most of the time. Over-commitment of resources can help in utilizing the infrastructure efficiently (higher consolidation and cost savings), but might result in overload situations which means more SLA violations (SLAVs) and hence lower quality of service (QoS). Demand-based placement utilizes a data center's PMs more efficiently but might lead to more SLAVs from PM overload and excessive VM migrations. Figure 1, depicts this trade-off between the efficient utilization of the PMs on one hand and the resulting SLAVs on the other hand.

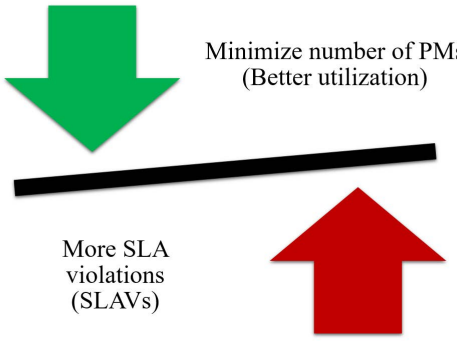

Figure 1: Trade-off between PM utilization and SLAVs

In this paper, we aim to utilize the cloud data center efficiently and flexibly in terms of a provider's goals by minimizing the number of running PMs while reducing SLA violations (SLAVs) and the overload per PM. This aim is achieved by exploiting the range between demand-based and reservation-based placement. We are adopting a new parameter-based dynamic VM consolidation solution which allocates resources in the range between the actual demand and the full reservation based on a single parameter (α). This parameter-based solution is controlled by the cloud provider and offers wide flexibility to choose an appropriate balance between efficient utilization and SLA violations based on the provider's optimization objectives.

The remainder of this paper is organized as follows. Section 2 categorizes some of the existing VM placement solutions. Section 3 describes the VM allocation problem. Sources of SLAVs in IaaS are summarized in Section 4. The proposed parameter-based VM placement solution is described in Section 5. Section 6 evaluates the proposed solution, while Section 7 concludes the paper.

2 RELATED WORK

An initial VM placement controller either accepts or rejects placement requests based upon resource availability; then assigns VMs to the appropriate PMs based on the deployed initial placement strategy. Dynamic placement reallocates VMs using live migration for achieving a placement that utilizes the PMs efficiently, which eventually helps in saving energy. We are analyzing the literature in three main areas namely *initial placement*, *reservation-based dynamic placement* and *demand-based dynamic placement*.

Initial VM Placement: The area of the initial VM placement is ripe enough. Shi Lei *et al.*[28] proposed two solutions for VM placement using integer linear programming (ILP) and a heuristic using using first fit decreasing (FFD). Beloglazov *et al.*[3] solved the initial placement of VMs by adopting a power-aware best fit decreasing algorithm. Rabbani *et al.*[27], explicitly consider the virtual switches and links in the placement of the virtual data center (VDC) to PMs. Quang-Hung *et al.*[26] solved the static VM allocation problem with the goal of minimizing energy consumption by proposing a heuristic solution and a genetic algorithm-based solution. Wuhib *et al.*[29] solved the initial VM allocation problem by proposing an algorithm inspired by Mitzenmacher's paper titled "*The power of two choices in randomized load balancing*" [20]. Rabbani *et al.*[27] propose a static virtual data centre (VDC) embedding solution that balances the load while considering communication costs.

Dynamic reservation-based VM Placement: Borgetto *et al.*[5] developed heuristics that dynamically allocate VMs in a Cloud data center with the goal of saving energy consumption while maintaining the resources required by users. Lucanin *et al.*[18] proposed the concept of grid-conscious Cloud model that is aware of the changes in electricity prices during the day. They proposed a scheduling algorithm, *peak pauser*, which pauses the green instances during the hours of peak electricity prices. Ferdaus *et al.*[13] modelled the VM consolidation problem as a multi-dimensional vector packing problem (mDVPP). They used an ant colony optimization (ACO) metheuristic to consolidate VMs in a cloud data center with the goal of minimizing energy consumption and resource wasting. The main focus was on balancing the resource utilization of servers across resource types.

Dynamic demand-based VM Placement: Gandhi *et al.*[14] propose a reactive and proactive approach for resource allocation that tries to minimize SLA violations and energy consumption in data centers. The proposed approach consists of four controllers, namely, *base workload forecaster*, *predictive controller*, *reactive controller* and the *coordinator*. Calcavecchia *et al.*[6] consider the dynamic nature, ongoing VM allocation and deallocation requests, of the VM placement problem. They propose a technique called Backward Speculative Placement (BSP) which assumes previous knowledge of historical workload traces of VMs in the Cloud data center. The BSP technique divides the VM placement into two phases. The first phase, continuous deployment, only deals with the newly received VM requests and it assigns VMs-to-PMs using a best fit decreasing (BFD) algorithm. The BFD algorithm places VMs according to a score function which measures the level of demand dissatisfaction. The second phase, ongoing optimization, aims at optimizing

VM placement by migrating VMs from most loaded PMs to others that are likely to ensure a better demand satisfaction. More *et al.*[22] propose dynamic heuristics that control changes in the cloud environment for optimizing energy utilization based on CPU utilization. They make use of soft scaling, DVFS, VM migration and consolidation and power switching techniques for energy saving. Beloglazov *et al.*[3] addressed the dynamic VM consolidation problem in cloud data centers for minimizing energy consumption and SLA violations. They followed a heuristic-based approach for dynamically consolidating VMs in the least possible number of host PMs. This approach followed a divide-and-conquer strategy where the main problem was divided into four subproblems, namely, *host overload detection*, *host underload detection*, *VM selection* and *VM placement*. Chowdhury *et al.*[7] improved Beloglazov *et al.*[3] work by clustering the VMs to be migrated and proposing a different set of VM placement algorithms. Monil [21] used fuzzy logic for a better VM selection of the VM selection subproblem in Beloglazov *et al.*[3]. Mosa *et al.*[23] propose a dynamic utility-based VM allocation solution that makes use of a genetic algorithm to maximize the profit of the cloud provider by minimizing energy consumption and SLA violations. Farahnakian *et al.*[12] used the Ant Colony meta-heuristic to consolidate VMs for energy and maintain the QoS requirements. Xiao *et al.*[30] propose a dynamic demand-based VM allocation solution that uses the skewness metric to improve the utilization of the servers.

3 THE VM ALLOCATION PROBLEM

The VM allocation problem relates to the mapping of VMs to PMs. To find a good trade-off between utilization and PM overload, we aim to minimize the underutilization and the overutilization for the data center's PMs. Section 3.1 introduces the mathematical formulation of the VM allocation problem and Section 3.2 defines the objective function used in this paper.

3.1 Mathematical formulation

Given a *Cloud Data Center* (CDC) having n physical machines (PMs) defined as $P = \{p_1, p_2, \ldots, p_n\}$; each $p \in P$ is defined by three resource types, namely CPU, RAM and bandwidth (BW). The capacity c of resource type r of PM_i is defined as $c_i^r \in C^r$, where $C^r = \{c_1^r, c_2^r, \ldots c_n^r\}$. Therefore, c_i^{cpu}, c_i^{ram} and c_i^{bw} represent PM_i's capacity of CPU, RAM and bandwidth, respectively. Any PM i at time t could be either *active (running)* or *inactive* (switched off or in any of the power saving modes).

$$p = \begin{cases} 0 & \text{if: PM is not active} \\ 1 & \text{if: PM is active} \end{cases}$$

Given m VMs, $v \in V$ where $V = \{v_1, v_2, \ldots v_m\}$, the VMs V are initially allocated to a subset of active PMs P using a mapping function, as shown in Figure 2, $f : V \rightarrow P$ such that $\forall v \in V, \exists p \in P : p = f(v)$. The adaptive consolidation reallocates VMs during each scheduling interval $t \in T$, $T = \{t_1, t_2, \ldots t_q\}$. The total demand, from the hosted VMs, for each resource type r of PM_i at time t is defined as $d_{it}^r \in D^r$, where $D^r = \{d_1^r, d_2^r, \ldots d_n^r\}$. Therefore, d_{it}^{cpu}, d_{it}^{ram} and d_{it}^{bw} represent the CPU, RAM and bandwidth demand of PM_i at time t.

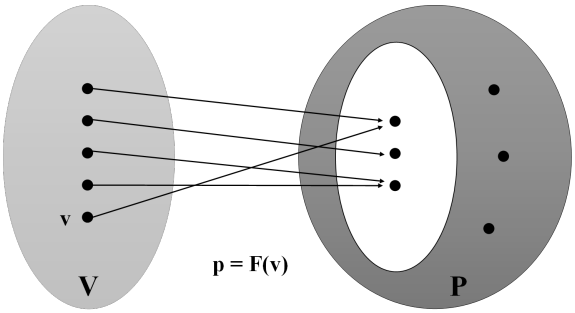

Figure 2: VMs-to-PMs mapping

Resource utilization defines the percentage of usage of each resource type; the utilization of resource r of PM_i at time t is $u_{it}^r \in U^r$, where $U^r = \{u_1^r, u_2^r, \ldots u_n^r\}$. The u_{it}^r is calculated by dividing the demand by total capacity as follows: $u_{it}^x = \frac{d_{it}^x}{c_{it}^x}$. Hence, utilization of CPU, RAM and bandwidth per PM_i at time t is represented by u_{it}^{cpu}, u_{it}^{ram} and u_{it}^{bw} respectively. A PM could be in one of these five utilization states; namely, *fully-utilized*, *partly-utilized*, *under-utilized*, *over-utilized* or *non-utilized*. A *non-utilized* PM means that it is either switched off, idle or in one of the power saving modes. A PM is considered as over-utilized when $d_{it}^r > c_{it}^r$. The utilization of PM i at time t defines the PM's state as follows, where *under_thr* and *over_thr* represent the underutilization and overutilization thresholds respectively.

$$p_{it} = \begin{cases} \text{non-utilized} & \text{if: } \sum_{r=1}^{z} u_{it}^r = 0\% \\ \text{fully-utilized} & \text{if: } u_{it}^r = 100\% \\ \text{partly-utilized} & \text{if: } under\_thr < u_{it}^r < over\_thr \\ \text{under-utilized} & \text{if: } 0\% < max_{r=1}^{z}(u_{it}^r) \leqslant under\_thr \\ \text{over-utilized} & \text{if: } max_{r=1}^{z}(u_{it}^r) \geqslant over\_thr \end{cases}$$

A PM is underutilized when the maximum utilization of any resource type is less than the underutilization threshold. The percentage of underutilization per resource r of PM_i at time t is $w_{it}^r \in W^r$, where $W^r = \{w_1^r, w_2^r, \ldots, w_n^r\}$. Accordingly, the underutilization of CPU, RAM, bandwidth per PM i at time t is w_{it}^{cpu}, w_{it}^{ram} and w_{it}^{bw} respectively. The $w_{it}^r = (100 - u_{it}^r)$ and $w_{it}^r = 0\%$ if $u_{it}^x \geqslant 100\%$.

The aggressive consolidation of VMs may result in overutilization per any resource type and hence violates the Quality of Service (QoS) requirements agreed through a service level agreement. The overutilization per resource r of PM_i at time t is defined as $o_{it}^r \in O_i^r$, where $o_{it}^r = \frac{d_{it}^r - c_{it}^r}{c_{it}^r}$. The CPU, RAM and bandwidth overutilization per PM i at time t is defined as o_{it}^{cpu}, o_{it}^{ram} and o_{it}^{bw} respectively.

The average underutilization of resource type r for all active PMs in a single scheduling interval t is shown in (1).

$$\bar{W}_t^r = \frac{1}{k} . \sum_{i=1}^{n} w_i^r \tag{1}$$

Here, k defines the number of currently active partly-utilized or underutilized PMs. Equation (2) calculates the average underutilization per resource r of PMs across all q scheduling intervals:

$$\bar{W}^r = \frac{1}{q}.\sum_{t=1}^{q} \bar{W}_t^r \tag{2}$$

Assuming that there are l overutilized (overbooked) PMs, the average resource overutilization for all the overutilized PMs in the scheduling interval t is computed in (3).

$$\bar{O}_t^r = \frac{1}{l}.\sum_{i=1}^{n} o_i^r \tag{3}$$

Equation (4) calculates the average resource overutilization per resource type r of all running PMs across all scheduling intervals.

$$\bar{O}^r = \frac{1}{q}.\sum_{t=1}^{q} \bar{O}_t^r \tag{4}$$

3.2 Objective Function

The objective function seeks to utilize the PMs efficiently (maximize the utilization) by minimizing the estimated underutilization per resource type, while reducing SLAVs by minimizing overutilization. Equation (5) defines the objective function supposing that there are z resource types.

$$\text{Minimize } (\sum_{r=1}^{z}(S_{rw}.\bar{W}_t^r + S_{ro}.\bar{O}_t^r)) \tag{5}$$

Here, $\bar{W}_t^r$ is the average underutilization per resource r of PMs in interval t. Minimizing the underutilization per PM inherently means reducing the number of running PMs and hence increase utilization of the CDC. The $\bar{O}_t^r$ is the average overutilization per resource r for PMs in interval t. Minimizing overutilization reduces the SLAVs resulting from the aggressive consolidation of the VMs. The $0 \leqslant S_{rw} \leqslant 1$ and $0 \leqslant S_{xo} \leqslant 1$ are weights representing the relative significance of the underutilization and overutilization of each resource r respectively.

4 SOURCES OF SLA VIOLATIONS IN IAAS

A Service Level Agreement (SLA) defines the Quality of Service (QoS) requirements in the form of a contractual document between the service provider and the customer [17]. An SLA violation (SLAV) means that the previously agreed-upon SLA requirements have not been met. The SLA requirements differ between various cloud service models. For example, in the software as a service (SaaS) model, the response time is an important requirement of an SLA. On the other hand, in the infrastructure as a service (IaaS) model, response time is not a main attribute; availability is a crucial requirement. We need to define an SLA independent of the workload. In this work, we say that the SLA requirements are met when 100% of the resources requested by a VM are delivered at any time bounded by the maximum resource capacity of the VM. There are many sources of SLA violations in IaaS clouds including *live migration* of VMs, *overutilization of the PMs*, system outage and interference among co-located workloads. However, we are considering only live migration and overutilization of the PMs. Overutilization of the PMs results from the aggressive consolidation of VMs through demand-based placement. Live migration causes performance degradation as the VM is not available during the migration time. The time required to migrate a VM is equal to the ratio of the total memory used by that VM divided by the available network bandwidth. The following subsections 4.1 and 4.2 describe violations from live migration and PM overutilization in more detail.

4.1 Violations due to live migration (SLAVM)

Live migration is a valuable tool for data centers' administrators as it aids in balancing the load between physical servers, fault management and low-level system maintenance [8]. The majority of the live migration process is completed while a VM is running, and hence it improves the efficiency of the data centers with minimum downtime. The process of copying pages is done while the VM is running. For example, if we are migrating VM_j from PM_1 to PM_2 then the downtime of VM_j is only the time required to suspend VM_j on PM_1 plus the time to generate ARP to redirect traffic to PM_2 in addition to the time to synchronize the state of VM_j on PM_2 [8]. The migration time depends on the available network bandwidth as well as the RAM capacity of the VM [9]. There is a linear relationship between RAM capacity and migration time. However, there is a negative correlation between migration time and network bandwidth. Equation (6) calculates the time required to migrate a VM j supposing that the image and data of VMs are stored on a storage area network (SAN) over a network.

$$t_{mj} = \frac{d_j^{ram}}{a_j^{bw}} \tag{6}$$

Here, t_{mj} is the time required to migrate VM j; d_j^{ram} and a_j^{bw} represent VM j's CPU demand and the available network bandwidth respectively. The number of VM migrations should be controlled in a way that helps strike a good balance between efficient utilization of the PMs and the resulting SLAVs due to migration. Too many migrations add an overhead and may degrade performance, while too few migrations may lead to less efficient utilization of the PMs: a balance is required to deal with this trade-off.

The metric SLAVM (standing for SLA violations from migrations) estimates the performance degradation resulting from the migration of VMs among PMs. We calculate *SLAVM* for a specific VM by dividing the *under-allocation per CPU due to migration* by the VM's *total CPU demand*. Equation (7) calculates the *SLAVM* for all VMs.

$$SLAVM = \frac{1}{m}\sum_{j=1}^{m}\frac{d_{dj}^{cpu}}{d_j^{cpu}}, \tag{7}$$

where d_{dj}^{cpu} is the VM_j's under-allocated CPU demand due to migration, d_j^{cpu} is the total CPU demand by VM_j and m is the total number of VMs.

4.2 Violations due to overutilization (SLAVO)

SLAs are satisfied when 100% of the VM's demand is met at any time. The SLAVO metric computes the SLA violations from PM's *overutilization*, which is the time during which the PM is 100% utilized in any of the resource types. SLAVO happens when the resource demand is greater than the allocated amount bounded by

the VM's capacity. The SLAVO of PM i is calculated as shown in Equation 8.

$$SLAVO_i = \frac{t_{oi}}{t_{ai}} \tag{8}$$

Here, t_{oi} is the time period during which PM i overutilized in one of its resource types (CPU, RAM or bandwidth) and t_{ai} is the total time during which PM i is active (running). Equation 9 computes the average SLAVO per all PMs in the CDC.

$$\text{SLAVO} = \frac{1}{N}\sum_{i=1}^{N}\frac{t_{oi}}{t_{ai}} \tag{9}$$

5 PARAMETER-BASED VM CONSOLIDATION

5.1 Action-based VM consolidation framework

Adaptive VM consolidation is an example of a *self-managing* process that adapts to changes in the CDC. Action-based, also called *rule-based*, policies are often used for creating such self-managing systems. Action-based systems take the form of *if* (condition) *then* (action) and only care about what action the system should take whenever it is in any given state. Action policies do not specify the state that the system will reach after taking the action; it is all about what the system should do in the current state.

A general action-based framework for VM consolidation has been adopted by [4, 21, 30]. This general framework divided the VM reallocation problem into four subproblems, namely, *cold PM detection*, *hot PM detection*, *VM selection* and *VM placement*. The state of the active PM can be *hot*, *cold* or *normal*. An active PM is hot if it is overutilized, cold if it is underutilized and normal if it is partly utilized (neither overutilized nor underutilized). Figure 3 exhibits an example of an action-based policy for VM consolidation.

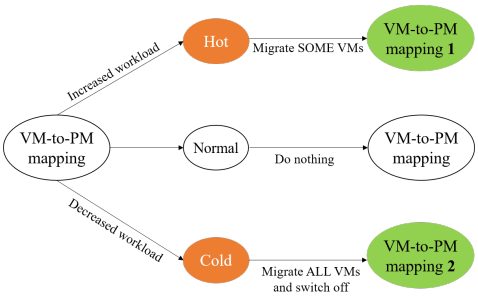

Figure 3: Action-based policy for VM consolidation

5.1.1 Cold PM detection. A PM is *cold*, when it is underutilized. Therefore, if the PM's *state* is *cold*, then the *action* should be *migrating all the hosted VMs* to other PMs, if possible, and switch that PM off or into one of the power saving modes.

5.1.2 Hot PM detection. A PM is *hot*, when it is overutilized which happens when the total demand is greater than the PM's capacity. If the PM's *state* is *hot*, then the *action* is *migrating some VMs* until the PM becomes normal; a PM is normal when it is not overutilized (*partly-utilized* as defined in Section 3.1). There are many algorithms that can be used for hot PM detection such as static threshold, local regression, robust local regression or median absolute deviation [4, 30]. We have used a static threshold that considers a PM as *hot* when resource utilization is greater than a static value (for example, 80%, 90%, 100%).

5.1.3 VM Selection. When a PM is detected as hot, then we need to migrate one or more VMs from that hot PM until it becomes normal. The VMs can be randomly selected, or can be selected based on specific criteria such as RAM or CPU utilization or using fuzzy logic as in [21]. Here, we have selected the VM with the minimum RAM demand to ensure minimum migration time per the VM.

5.1.4 VM Placement. VM placement is responsible for mapping VMs to PMs to achieve the cloud provider objectives without violating the constraints; using initial and dynamic placement. *Initial VM placement* accepts new VM placement requests and allocates VMs to PMs based on the resource capacity defined by the VM types. Initial VM placement is a kind of N-dimensional bin packing problem where PMs represent the bins, VMs represent the items to be packed and bin size is defined by the capacity of the resource types. On the other hand, *dynamic VM placement* starts with an existing VM-to-PM mapping and tries to generate a better mapping, if possible, based on the current one. The dynamic placement tries to find suitable PMs for allocating the VMs selected from hot and cold PMs. Finding a suitable PM for VM migration is crucial as it greatly affects the efficiency of the whole VM consolidation solution.

The dynamic reallocation of the VMs runs periodically for finding a better allocation that meets the cloud provider's objectives. The dynamic demand-based placement of VMs results in SLAVs, while the reservation-based does not utilize the data center efficiently. Therefore, we have introduced *parameter-based* VM placement that enables the cloud provider to fine tune the trade-off between utilization and SLA violations. The *parameter-based* VM placement allocates resources per VM (in the range between demand and total reservation) as shown in Equation 10.

$$\text{para-based}^r_{dcj} = \alpha.(c^r_j - d^r_j) + d^r_j\,. \tag{10}$$

Here $\text{para-based}^r_{dcj}$ is the new amount of resources, *between reservation and demand)*, that needs to be allocated for VM_j, c^r_j is the capacity of resource r per VM_j (reservation) and d^r_j represents the current demand. The parameter α specifies the desired scale between demand and reservation. When $\alpha = 0$ this means that we are adopting demand-based placement, while $\alpha = 1$ corresponds to reservation-based placement. Our proposed solution works as a knob in the range $0 \leqslant \alpha \leqslant 1$ as shown in Figure 4.

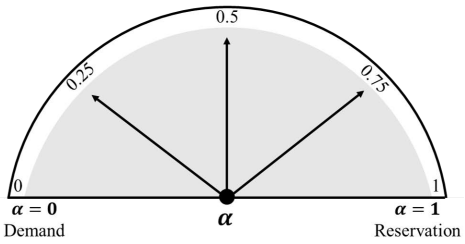

Figure 4: Parameter-based solution uses $0 <= \alpha <= 1$

Algorithm 1 describes the proposed parameter-based VM placement solution which works for both initial and dynamic placement based on the "*type*" parameter; "*type*" is either *"initial"* or *"dynamic"*. The parameter-based VM algorithm works on the range available between reservation and demand. In Algorithm 1, VMs are sorted decreasingly based on the VMs' CPU capacity when the *type* is "*initial*" or based on the CPU demand when *type* is "*dynamic*". Active and inactive PMs are sorted increasingly based on the PM's available CPU. Then, the algorithm tries to map each VM to a suitable PM. For the initial placement (type="initial"), the $isSuitable(vm, type, activePm)$ method returns true if the available CPU per the PM is greater than VM's CPU capacity. For dynamic placement (type="dynamic"), the $isSuitable(vm, type, activePm)$ method returns true when the PM's available CPU is greater than or equal to para-based$^{r}_{dcj}$ which is defined in Equation 10. The algorithm switches one of the suitable inactive PMs on, when there is no suitable active PM. The $isSuitable(vm, type, inactivePm)$ returns true if the inactive PM is suitable for hosting the VM.

Algorithm 1 Parameter-based virtual machine placement

```
1:  vmList ← sort(vmList, type, "DESC");
2:  inactivePMs ← sort(inactivePMs, "available^cpu", "ASC");
3:  mapping ← Empty;
4:  for all (vm in vmList) do
5:      activePMs ← sort(activePMs, "available^cpu", "ASC");
6:      for all (activePm in activePms) do
7:          if (isSuitable(vm, type, activePm)) then
8:              mapping.add(vm, activePm)
9:      if the vm is still not allocated then
10:         for all (inactivePm in inactivePms) do
11:             if (isSuitable(vm, type, inactivePm)) then
12:                 mapping.add(vm, inactivePm)
13: return mapping;
```

6 EXPERIMENTAL EVALUATION

The following experiments aim to explore the range between the actual demand and the full reservation using the proposed parameter-based VM placement solution. The experiments test the effect of changing the allocated CPU resources by tuning the value of α, previously defined in Equation (10). The allocated resources range from the demand-based ($\alpha = 0$) to reservation-based ($\alpha = 1$) allocation.

6.1 Settings of the simulation environment

The experiments were conducted with the CloudSim[6] simulation toolkit using four VM types and two PM types. For the evaluation of the proposed parameter-based solution, we have tested three types of workload, namely, *random*, *normal PlanetLab traces* and *skewed PlanetLab traces*, which are discussed in Section 6.1.1.

For the conducted experiments: for *hot PM detection*, we have used a static overutilization threshold of CPU utilization. For example, a PM is considered as hot, when CPU utilization is 100%. For *VM selection*, we have chosen the VM with the minimum RAM. The simulation works for a whole day and dynamic reallocation takes place every 5 minutes of simulated time. The Cloud Data Center (CDC) consists of 1033 VMs and 800 PMs.

6.1.1 Workload.

Randomly generated workload data: We randomly generate workload data representing CPU utilization every 5 minutes based on a uniform random distribution with values in the range 0 to 1. This workload represents the list of jobs or tasks assigned to VMs expressed in percentage of CPU utilization. Figure 5 (a) shows the average utilization for the random workload.

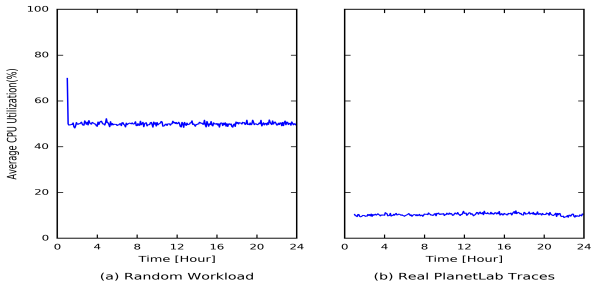

Figure 5: Average CPU utilization for 1033 VMs of Random and PlanetLab workload

Real PlanetLab traces: Real workload traces from the CoMon project[7] which generates monitoring statistics about the PlanetLab[8] platform were used. The workload data represent CPU utilization recorded each 5 minutes collected in the period between March and April 2011. Figure 5 (b) shows the average CPU utilization of the 1033 VMs on the 20^{th} of April 2011. The average utilization is less than 13% which ensures the fact that servers are underutilized most of the time. The average utilization does not demonstrate the change in workload over time, so we have selected two VMs, from the PlanetLab machines, to show the variations in workload as shown in Figure 6.

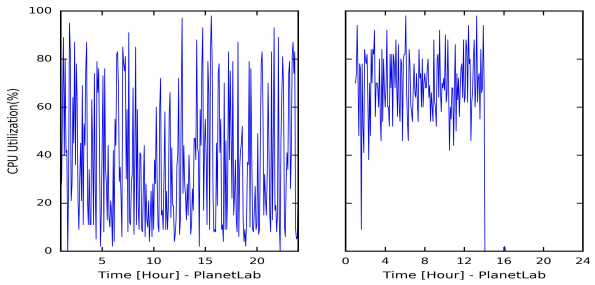

Figure 6: CPU utilization of 2 selected PlanetLab machines

Skewed PlanetLab traces: As the normal PlanetLab traces have low utilization, we created another workload, which skews the original PlanetLab workload multiplying it by 5 bounded by the VM size.

[6] http://www.cloudbus.org/cloudsim/

[7] http://comon.cs.princeton.edu/

[8] https://www.planet-lab.org/

Table 1: Energy consumption (in Watts) at different CPU utilization levels [4]

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

6.1.2 Performance Metrics.

(1) *Energy consumption:* CloudSim uses the SPECpower[9] benchmark for energy consumption. SPECpower calculates energy consumption based on the current CPU utilization. Table 1, from [4], shows the energy consumption, in Watts, at different utilization levels.

(2) *Absolute normalized number of running PMs (NPMs)*:
NPMs counts the number of active PMs in the CDC considering both the *PMs' capacity* (defined by CPU capacity) and the PM's *running time*. As an illustration, if there is a PM running during the whole simulation time and its CPU capacity is equal to the capacity of the largest PM, then *NPMs* will be increased by 1. Equation 11 computes the normalized number of PMs.

$$\text{NPMs} = \sum_{i=1}^{n} (norm_i^{cpu}.norm_i^{ta}) \tag{11}$$

Here, $norm_i^{cpu}$ is the normalized CPU size (in MIPs) of PM$_i$ and $norm_i^{ta}$ is the normalized active time of PMi. Equation (12) computes $norm_i^{cpu}$ and $norm_i^{ta}$.

$$norm_i^{cpu} = \frac{c_i^{cpu}}{c_{max}^{cpu}}; \text{and } norm_i^{ta} = \frac{t_{ai}}{t_s} \tag{12}$$

Here, c_i^{cpu} is the CPU capacity of PM$_i$ and c_{max}^{cpu} is the CPU capacity of the largest PM in the CDC. Moreover, t_{ai} is the running time of PM$_i$ and t_s is the simulation time.

(3) *Number of migrations and migration Time (RT_m)*:
The *number of migrations* counts the total number of VM migrations taken place in the CDC. The relative migration time RT_m is the percentage of time required for all VMs' migrations which is calculated in Equation 13, where t_{mj} is the time required to migrate VM$_j$ and t_{aj} is the total time during which VM$_j$ is active.

$$RT_m = \frac{\sum_{j=1}^{m} t_{mj}}{\sum_{j=1}^{m} t_{aj}} \tag{13}$$

(4) *SLA violations (SLAV)*
SLAV, introduced in [4], is a composite metric that considers two sources of violations; from *overutilization* and VM *migrations* as shown in Equation (14).

$$SLAV = SLAVO.SLAVM \tag{14}$$

$SLAVO$ and $SLAVM$ were previously introduced in Sections 4.1 and 4.2, respectively.

(5) *Total number of SLA violations (NSLAVs)*:
There is a violation per PM$_i$ at time t, when the allocated CPU a_{it}^{cpu} is less than the demand $d^{cpu}s_{it}$ regardless of the time of the violation.

$$NSLAV_{it} = \begin{cases} 1 & \text{if: } a_{it}^{cpu} < d_{it}^{cpu} \\ 0 & \text{otherwise} \end{cases}$$

(6) *Average CPU utilization*:
The CPU utilization of PM$_i$ at time t is calculated as follows:

$$u_{it}^{cpu} = \begin{cases} \frac{a_{it}^{cpu}}{c_{it}^{cpu}} & \text{if: } a_{it}^{cpu} <= d_{it}^{cpu} \\ \frac{d_{it}^{cpu}}{c_{it}^{cpu}} & \text{otherwise} \end{cases}$$

6.2 Experimental Results

6.2.1 Experiment 1: using random workload.

Absolute number of PMs (NPMs) to number of SLA violations (NSLAVs): Figure 7 (a) shows the change in the number of SLAVs (NSLAVs) to the absolute number of PMs as α changes from 0 to 1. Each point in the figure represents the value of α defined in Equation 10. At the beginning, a small increase in α results in a clear and gradual decrease in the number of SLAVs. Using *demand-based* placement, $\alpha = 0$, the number of PMs is reduced by around 50% compared to the *reservation-based* placement, $\alpha = 1$. However, demand-based placement introduced SLA violations. The cloud provider can reduce more than 37% of the number of SLA violations with around 13% increase in the number of PMs when $\alpha = 0.2$.

Energy consumption to percentage of SLAV:. Figure 7 (b) demonstrates that the cloud provider can decrease energy consumption, in case of reservation-based, by around 28% while having a near 0% SLAV when the value of $\alpha = 0.6$. This enables the cloud provider to find a suitable trade-off between the cost of the provided service (number of PMs and energy) and the related violation costs.

Average CPU utilization: The main idea behind the reallocation of the VMs is to efficiently utilize the PMs by increasing the utilization. Demand-based placement ($\alpha = 0$) improves CPU utilization by 50% (70.7% compared to 47% in case of reservation-based). Figure 7 (c) shows that using values of α in the range [0.1, 0.3] improves PMs' utilization. Moreover, Figure 7 (c), also shows that the highest CPU utilization is achieved when the values of α was between 10% and 20%. This suggests that increasing the number of running PMs does not only reduce SLAVs, but it can sometimes improve the PM's utilization.

Number of VM migrations: Figure 7 (d) demonstrates that increasing the value of α between 0 and 1 will decrease the number of migrations. At $\alpha = 1$, *full reservation*, the number of migrations is zero.

[9] http://www.spec.org/power-ssj2008

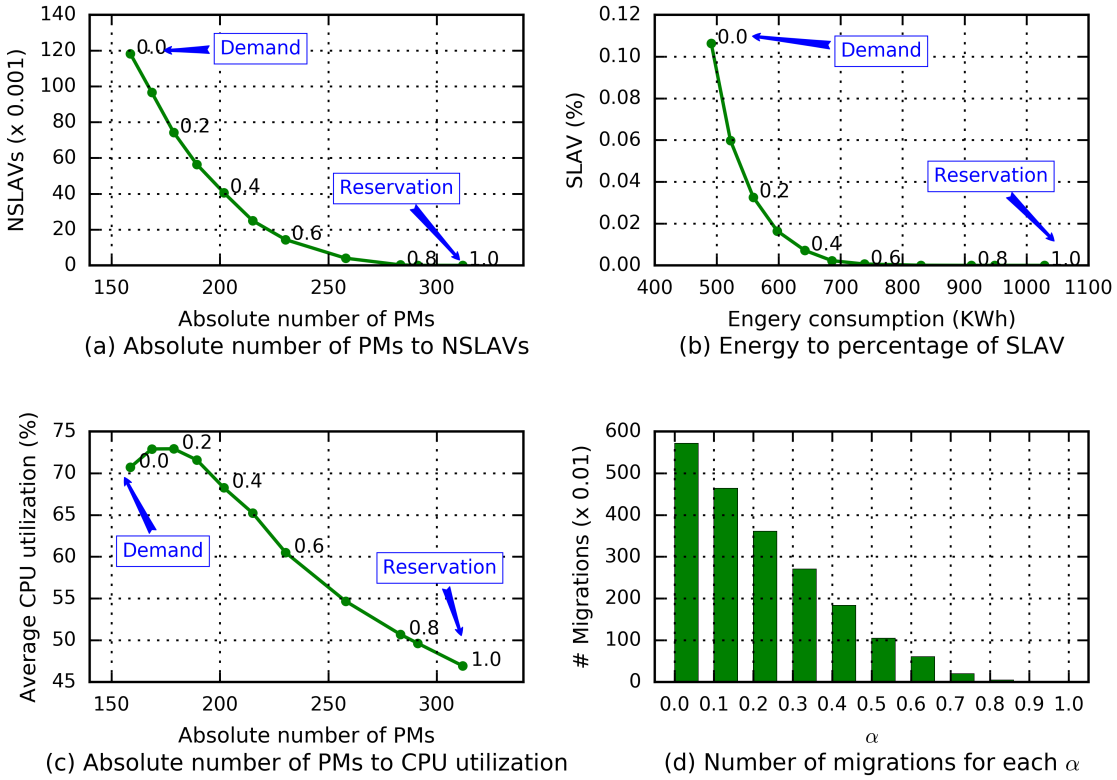

Figure 7: Experiment 1: using random workload

6.2.2 *Experiment 2: using normal PlanetLab traces.*

Absolute number of PMs (NPMs) to number of SLA violations (NSLAVs): The PlanetLab workload is of very low utilization and this has an effect on the results. Figure 8 (a) shows the NSLAVs to the absolute number of PMs as the value of α changes from 0 to 1. Figure 8 (a) exhibits a clear decrease in the number of SLA violations as the value of α increases from 0 to 0.2. There is also a non-significant increase in the NSLAVs at $\alpha = 0.5$ and 0.7.

Energy consumption to percentage of SLAV:. Again, increasing the value of α from 0.0 to 0.2 shows a clear decrease in the percentage of SLAVs as shown in Figure 8 (b). Compared to reservation-based placement ($\alpha = 1$), using $\alpha = 0.2$ reduces the percentage of SLAV to almost 0% (exactly 0.00005%), while reducing the energy consumption by around 66%.

Average CPU utilization: Figure 8 (c) shows the average CPU utilization to the absolute number of PMs as we move between demand-based and reservation-based placement. For the normal PlanetLab workload, it did not show exactly the same pattern that appeared in the random workload experiment. This is due to the nature of the PlanetLab workload, as the actual utilization of many VMs is 0%. Going back to Equation (10), when d_j^r is zero, the value of para-based$_{dcj}^r$ will always be a percentage of the reservation.

Number of VM migrations: Figure 8 (d) demonstrates that increasing the value of α between 0 and 1 will significantly decrease the number of VM migrations. At $\alpha = 1$, the number of migrations is 399: this is because of many PlanetLab VMs have 0% utilization which makes some PMs underutilized and hence migration happens.

6.2.3 *Experiment 3: using skewed PlanetLab traces.*

Absolute number of PMs (NPMs) to number of SLA violations (NSLAVs): Figure 9 (a) demonstrates that the skewed PlanetLab workload shows a similar behaviour with the random workload as the number of SLAVs decreases as we increase the value of α.

Energy consumption to percentage of SLAV:. For the skewed PlanetLab traces, the percentage of SLAVs decreases as we increase the allocated resources (as α increases). Figure 9 (b) shows that there is also a significant decrease in the percentage of SLAVs as α approaches 0.2 which means that it is a sort of a potential profit for the cloud provider to make use of it.

Average CPU utilization: Figure 9 (c) shows that the cloud provider could have better utilization if the provider increased the allocated workload (at $\alpha = 0.1$). Moreover, at $\alpha = 0.2$, the cloud provider could almost get the same utilization as in demand-based placement while reducing the number of SLAVs by 58% with only around 15% increase in the number of running PMs.

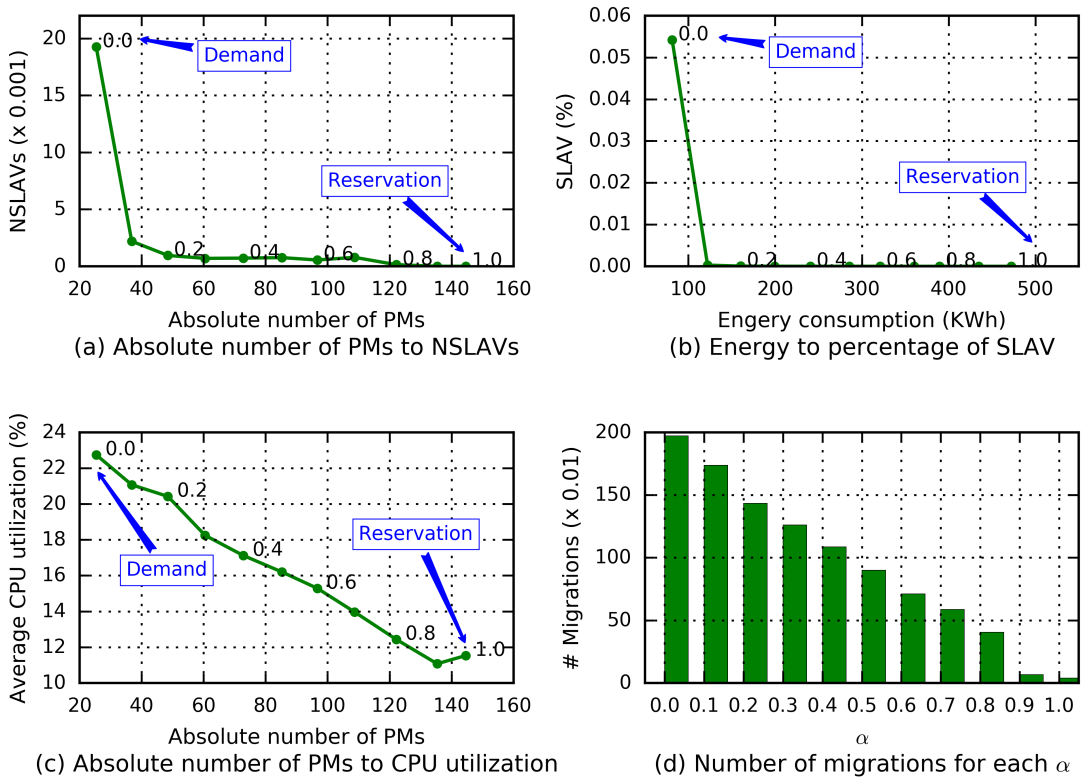

Figure 8: Experiment 2: using normal PlanetLab traces

Number of VM migrations: Again, as it was the case with random and normal PlanetLab traces, the skewed PlanetLab traces in Figure 9 (c) show that increasing the value of α from 0 to 1 will significantly decrease the number of VM migrations. Note that the number of migrations at $\alpha = 1$ does not equal to zero because some PMs are underutilized, as VM utilization is 0%, and hence some migration happens to improve the utilization of the PMs.

7 CONCLUSION AND FUTURE WORK

We have proposed a new parameter-based VM placement solution that works in the range between the current workload demand and the full reservation. This parameter-based VM placement solution can give the cloud provider more flexibility in choosing between the different trade-offs. The experiments show that, for some scenarios, increasing the number of running PMs results in better utilization of the CDC resources, in addition to the expected decrease in the number of SLA violations. Ongoing work will consider adaptive increase in the allocated resources and *dynamic change of the α values based on the history of the VMs*. Moreover, we will consider the proposed solution with multiple resource types (CPU, RAM and bandwidth).

ACKNOWLEDGMENTS

The first author would like to acknowledge the support of the Egyptian Government during his PhD program.

REFERENCES

[1] Luiz André Barroso and Urs Hölzle. 2007. The case for energy-proportional computing. (2007).
[2] L. A. Barroso and U. Hölzle. 2007. The Case for Energy-Proportional Computing. *Computer* 40, 12 (2007), 33–37. DOI: http://dx.doi.org/10.1109/MC.2007.443
[3] A Beloglazov, J Abawajy, and R Buyya. 2012. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems* (2012). http://www.sciencedirect.com/science/article/pii/S0167739X11000689
[4] Anton Beloglazov and Rajkumar Buyya. 2012. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience* 24, 13 (2012), 1397–1420.
[5] Damien Borgetto and Patricia Stolf. 2014. An energy efficient approach to virtual machines management in cloud computing. In *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE, 229–235.
[6] Nicolò Maria Calcavecchia, Ofer Biran, Erez Hadad, and Yosef Moatti. 2012. VM placement strategies for cloud scenarios. *Proceedings - 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012* (2012), 852–859. DOI: http://dx.doi.org/10.1109/CLOUD.2012.113
[7] Mohammed Rashid Chowdhury, Mohammad Raihan Mahmud, and Rashedur M Rahman. 2015. Implementation and performance analysis of various VM placement strategies in CloudSim. *Journal of Cloud Computing* 4, 1 (2015), 1.
[8] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. 2005. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. USENIX Association, 273–286.
[9] Waltenegus Dargie. 2014. Estimation of the cost of vm migration. In *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on*. IEEE, 1–8.
[10] Marcos Dias De Assuncao, Jean-Patrick Gelas, Laurent Lefevre, and Anne-Cécile Orgerie. 2012. The Green Gridfi5000: Instrumenting and using a Grid with energy sensors. In *Remote Instrumentation for eScience and Related Aspects*. Springer, 25–42.

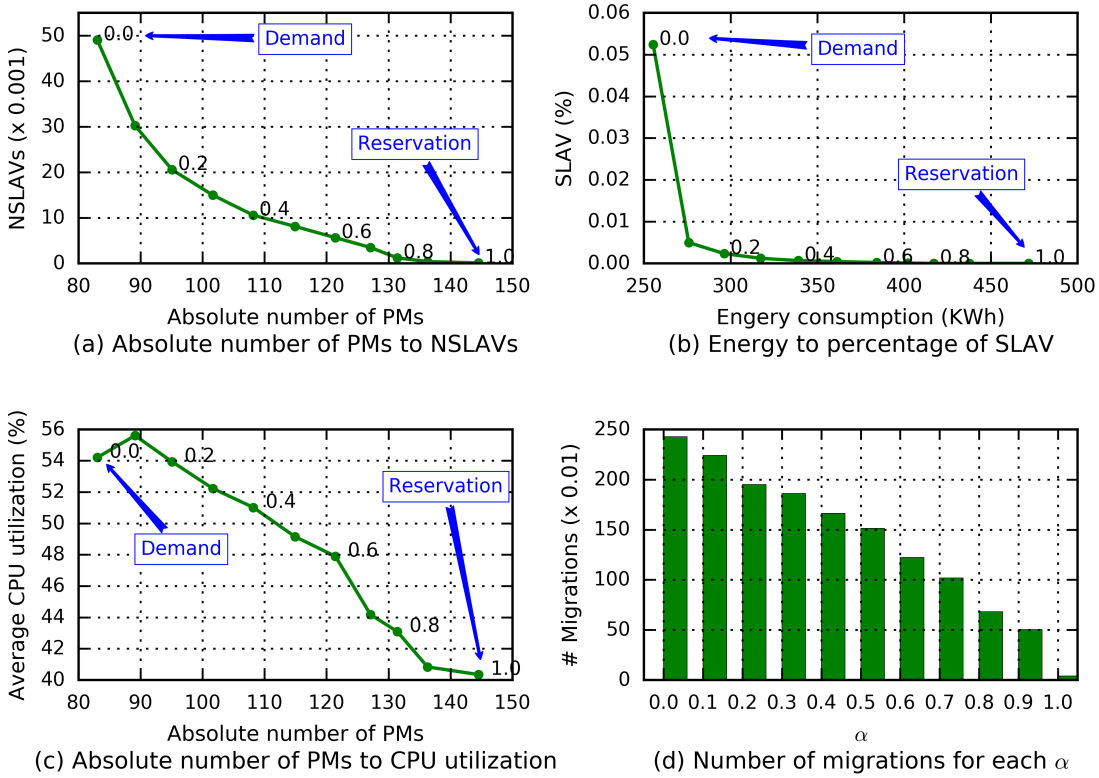

Figure 9: Experiment 3: using skewed PlanetLab traces

[11] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. 2007. Power provisioning for a warehouse-sized computer. In *ACM SIGARCH Computer Architecture News*, Vol. 35. ACM, 13–23.

[12] Fahimeh Farahnakian, Adnan Ashraf, Tapio Pahikkala, Pasi Liljeberg, Juha Plosila, Ivan Porres, and Hannu Tenhunen. 2015. Using ant colony system to consolidate vms for green cloud computing. *IEEE Transactions on Services Computing* 8, 2 (2015), 187–198.

[13] Md Hasanul Ferdaus, M Manzur Murshed, Rodrigo N Calheiros, and Rajkumar Buyya. 2014. Virtual Machine Consolidation in Cloud Data Centers Using ACO Metaheuristic.. In *Euro-Par*. 306–317.

[14] Anshul Gandhi, Yuan Chen, Daniel Gmach, Martin Arlitt, and Manish Marwah. 2012. Hybrid resource provisioning for minimizing data center SLA violations and power consumption. *Sustainable Computing: Informatics and Systems* 2, 2 (2012), 91–104. DOI: http://dx.doi.org/10.1016/j.suscom.2012.01.005

[15] Abdul Hameed, Alireza Khoshkbarforoushha, Rajiv Ranjan, Prem Prakash Jayaraman, Joanna Kolodziej, Pavan Balaji, Sherali Zeadally, Qutaibah Marwan Malluhi, Nikos Tziritas, Abhinav Vishnu, and others. 2016. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing* 98, 7 (2016), 751–774.

[16] A Hios and T Ulichnie. 2013. *Top 10 Data Center Business Management Priorities for 2013 about the Uptime Institute Network*. Technical Report. Technical Report, Uptime Institute.

[17] Craig A Lee and Alan F Sill. 2014. A design space for dynamic service level agreements in OpenStack. *Journal of Cloud Computing* 3, 1 (2014), 17.

[18] Drazen Lucanin and Ivona Brandic. 2013. Take a break: cloud scheduling optimized for real-time electricity pricing. In *Cloud and Green Computing (CGC), 2013 Third International Conference on*. IEEE, 113–120.

[19] Ching Chuen Teck Mark, Dusit Niyato, and Tham Chen-Khong. 2011. Evolutionary optimal virtual machine placement and demand forecaster for cloud computing. *Proceedings - International Conference on Advanced Information Networking and Applications, AINA* (2011), 348–355. DOI: http://dx.doi.org/10.1109/AINA.2011.50

[20] Michael Mitzenmacher. 2001. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems* 12, 10 (2001), 1094–1104. DOI: http://dx.doi.org/10.1109/71.963420

[21] Mohammad Alaul Haque Monil and Rashedur M Rahman. 2016. VM consolidation approach based on heuristics fuzzy logic, and migration control. *Journal of Cloud Computing* 5, 1 (2016), 1–18.

[22] Devwrat More, Sharad Mehta, Pooja Pathak, Lokesh Walase, and Jibi Abraham. 2014. Achieving Energy Efficiency by Optimal Resource Utilisation in Cloud Environment. In *Cloud Computing in Emerging Markets (CCEM), 2014 IEEE International Conference on*. IEE, 1–8.

[23] Abdelkhalik Mosa and Norman W Paton. 2016. Optimizing virtual machine placement for energy and SLA in clouds using utility functions. *Journal of Cloud Computing* 5, 1 (2016), 17.

[24] Nrdc.org. 2015. America's Data Centers Consuming and Wasting Growing Amounts of Energy. (2015). http://www.nrdc.org/energy/data-center-efficiency-assessment.asp

[25] Ilia Pietri and Rizos Sakellariou. 2016. Mapping virtual machines onto physical machines in cloud computing: A survey. *ACM Computing Surveys (CSUR)* 49, 3 (2016), 49.

[26] Nguyen Quang-Hung, Pham Dac Nien, Nguyen Hoai Nam, Nguyen Huynh Tuong, and Nam Thoai. 2013. A genetic algorithm for power-aware virtual machine allocation in private cloud. In *Information and Communication Technology-EurAsia Conference*. Springer, 183–191.

[27] Md Golam Rabbani, Rafael Pereira Esteves, Maxim Podlesny, Gael Simon, Lisandro Zambenedetti Granville, and Raouf Boutaba. 2013. On tackling virtual data center embedding problem. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE, 177–184.

[28] Lei Shi and Bernard Butler. 2013. Provisioning of requests for virtual machine sets with placement constraints in IaaS clouds. *IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)* (2013), 499–505. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6573024

[29] Fetahi Wuhib, Rerngvit Yanggratoke, and Rolf Stadler. 2013. Allocating Compute and Network Resources Under Management Objectives in Large-Scale Clouds. *Journal of Network and Systems Management* (2013), 1–26. DOI: http://dx.doi.org/10.1007/s10922-013-9280-6

[30] Zhen Xiao, Weijia Song, and Qi Chen. 2013. Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE transactions on parallel and distributed systems* 24, 6 (2013), 1107–1117.