Utilization Prediction Aware VM Consolidation Approach for Green Cloud Computing

Fahimeh Farahnakian, Tapio Pahikkala, Pasi Liljeberg, Juha Plosila, Hannu Tenhunen
Department of Information Technology, University of Turku, Turku, Finland.
fahfar@utu.fi, aatapa@utu.fi, pakrli@utu.fi, juplos@utu.fi, hanten@utu.fi

***Abstract*—Dynamic Virtual Machine (VM) consolidation is one of the most promising solutions to reduce energy consumption and improve resource utilization in data centers. Since VM consolidation problem is strictly NP-hard, many heuristic algorithms have been proposed to tackle the problem. However, most of the existing works deal only with minimizing the number of hosts based on their current resource utilization and these works do not explore the future resource requirements. Therefore, unnecessary VM migrations are generated and the rate of Service Level Agreement (SLA) violations are increased in data centers. To address this problem, our VM consolidation method which is formulated as a bin-packing problem considers both the current and future utilization of resources. The future utilization of resources is accurately predicted using a k-nearest neighbor regression based model. In this paper, we investigate the effectiveness of VM and host resource utilization predictions in the VM consolidation task using real workload traces. The experimental results show that our approach provides substantial improvement over other heuristic algorithms in reducing energy consumption, number of VM migrations and number of SLA violations.**

***Keywords*-Dynamic VM consolidation; bin-packing; utilization prediction model; energy-efficiency; SLA**

I. Introduction

Virtualization has become a powerful technology in data centers and cloud computing [1]. This technology allows a physical server or host to be shared between different virtual machines by Virtual Machine Monitor (VMM) or hypervisor to increase resource utilization. One of the main benefits of virtualization is Dynamic VM Consolidation (DVMC) that is used by data centers to reduce energy consumption. Energy efficiency is achieved by consolidating VMs into the minimum number of hosts and switching idle hosts to the sleep mode. DVMC uses live migration [2] ability that is provided by virtualization technology for transferring a running VM from one host to another. Migrating a VM can be advantageous either when a host is highly under-utilized, or when it is over-utilized. Therefore, resource management polices can be more flexible by migration operations. Nevertheless, live migration has a negative impact on the performance of applications running in a VM during a migration [3]. As providing the desired level of Quality of Service (QoS) between cloud providers and their users is critical, DVMC should minimize the number of migrations. Moreover, the aggregated total resource consumption of co-located VMs should be considered by DVMC in order to preserving QoS requirements. QoS requirements are commonly formalized in the form of Service Level Agreements (SLAs) that specify the required performance levels in terms of minimum throughput and maximum response time of the system.

DVMC problem can be formulated as a bin-packing problem which is NP-hard. As the bin-packing deals only with minimizing the number of bins used given a set of items with, it cannot apply for VM consolidation directly [4]. Great number of algorithms have been proposed to solve the VM consolidation as a bin-packing problem. However, they produce large amounts of unnecessary migrations and increase the risk of SLA violations. In contrast to most of the existing works that only minimize the number of hosts, this paper proposes a heuristic algorithm to minimize the number of VM migrations and SLA violations. Our algorithm performs DVMC in two phases: (1) migrating all VMs from least-loaded hosts to most-loaded hosts (2) migrating some VMs from the hosts that are overloaded currently or become overloaded in the near future. Moreover, the proposed DVMC allocates a VM to a host based on the current and future resource requirements. Based on the obtained results in our previous works [5] [6], the prediction model based on the K-Nearest Neighbor Regression (K-NNR) can predict the future resource utilization better than the linear regression. Therefore, we use KNNR in order to predict the future resource utilization of host in this paper. To train and test of the prediction model, we generate the historical data by running various real workload data that presents an IaaS cloud environment such as Amazon EC2. In addition, we investigate the prediction accuracy of model by evaluating accuracy of predictions made with partial data. Experimental results on the real workload traces show that the proposed DVMC can reduce the energy consumption and the number of VM migrations while maintains required performance levels in data centres.

The remainder of this paper is organized as follows. Section II discusses some of the most important related works and briefly reviews some heuristic algorithms for solving bin-packing problem. We describe the problem statement in Section III. Section IV and Section V present the system architecture and the proposed dynamic VM consolidation approach, respectively. Section VI describes the experimental design, setup and the experimental results. Finally, we present our conclusions in Section VII.

2159-6190/15 $31.00 © 2015 IEEE
DOI 10.1109/CLOUD.2015.58

II. Related Work and Background

Dynamic VM consolidation techniques have been popularly used for increasing resource utilization and energy-efficiency in data centers. The VM consolidation has been well studied in the literature [7], [8], [4], [3], [9]. Bobroff et al. [7] present a dynamic server migration and consolidation algorithm to reduce the amount of physical capacity required to support a specific rate of SLA violations for a given workload. In their algorithm, bin-packing heuristic and time series forecasting techniques are combined to minimize the number of hosts required to run a workload. However, their algorithm does not consider the number of migrations needed to a new placement. In [8], ant colony system is used to to find a near-optimal solution. They defined a multi-objective function that take into account both the number of dormant hosts and the number of migrations. Secron [4] assumes a threshold value to prevent CPU's host from reach 100% utilization that leads to performance degradation. Therefore, it tries to keep the total usage of a host below the threshold value. However, setting static thresholds are not efficient for an environment with dynamic workloads, in which different types of applications may run on a host. Therefore, the threshold value should be tuned for each workload type and level to allow the consolidation task to perform efficiently. Beloglazov and Buyya [3] proposed adaptive upper and lower thresholds based on the statistical analysis of the historical data. A multi-resource VM placement algorithm was proposed in [10] for maximizing the resource utilization and balancing the load across different types of resources in cloud data centers.

The VM consolidation problem can be formulated as a bin-packing problem. A bin-packing problem is to put a number of items into a finite number of bins so that the minimal bins are used. so that each VM is considered as an item and each host is assumed as a bin in the VM consolidation problem. As the bin-packing problem is known to be NP-hard, there are many heuristic algorithms to solve it. Among the most popular heuristic algorithms, First Fit (FF) algorithm which places each item into the first bin in where it will fit. The second popular heuristic algorithm is the Best Fit (BF) which puts each item into the filled bin in which is fits. Moreover, the FF and BF heuristics can be improved by applying a specific order of items such as First Fit Decreasing (FFD) and Best Fit Decreasing (BFD) algorithms. Nevertheless, such classical algorithms cannot be used directly for VM consolidation and they should be modified for applying to the VM consolidation task for these reasons.

First, VMs and hosts are characterized with multi dimensional resources such as CPU, memory, and network bandwidth. For example the VM consolidation becomes a typical 2D-bin-packing problem with considering CPU and memory constraints. Second, VM consolidation problem can be modeled as a bin-packing problem with different bin (host) sizes unlike the classical bin-packing problem where bin capacities are equal. Third, classical bin-packing algorithms only rely on minimizing the number of bins (single-objective), although we should consider other objectives such as the number of migrations and SLA violations to design an reasonable solution to solve VM consolidation. In [3], authors have presented a modified version of the BFD algorithm for the VM placement and have reported substantial energy saving based on simulation-driven results. PMapper [9] applies a modified version of the FFD heuristic to perform a power and migration cost aware server consolidation. Similarly in [11], a framework called EnaCloud is presented where a modified version of the BF algorithm is used for dynamic application placement. This work only focuses on reducing energy consumption and do not consider SLA violations. Moreover, Secron [4] has been modified FF and BF algorithms in order to minimize the number of hosts without compromising their performance. In this paper, we propose a modified BFD algorithm, named Utilization Prediction-aware Best Fit Decreasing (UP-BFD) where the main contributions of the paper are summarized as follow:

- UP-BFD produces an acceptable solution for the dynamic VM consolidation problem. It finds the optimal trade-off between energy saving and SLA violations.
- In order to minimize the number of migrations, UP-BFD allocates a VM to a host according to the current and future resource requirements.
- In order to reduce SLA violations, UP-BFD migrates some VMs from the hosts that are currently overloaded or become overloaded such nearly.
- UP-BFD enables proactive consolidation of VMs using a resource utilization prediction model. This model uses the K-nearest neighbor regression [6] to predict CPU utilization of VMs and hosts based on the historical data.
- We present comparative results by conducting a performance evaluation study of various utilization prediction based consolidation techniques using real workload traces.
- We study the relationship between energy consumption, the number of migrations and SLA violations based on different CPU utilization thresholds. The value of the threshold ranges from 50% to 100%.

III. Problem Statement

An efficient VM consolidation method optimizes the VMs placement in order to minimize the number of active hosts with maximum expected benefit. This benefit is the combination of two important factors: the rate of SLA violations and the number of VM migrations. We can increase these benefits in advance by allocating VMs to hosts based on their future resource utilization. To describe the problem statement of a conventional VM consolidation method without the prediction of future resources, assume two examples that are shown in Figure 1 and 2. We have two hosts and three VMs are allocated on hosts. In Figure 1(a), the CPU utilization of $Host1$ and $Hos2$ are 0.35 and 0.60, respectively. As $Host1$ has enough resources to allocate $VM3$, a conventional VM consolidation migrates $VM3$ to $Host1$ for reducing the number of active hosts and switching $Host2$ to the sleep mode. At the time

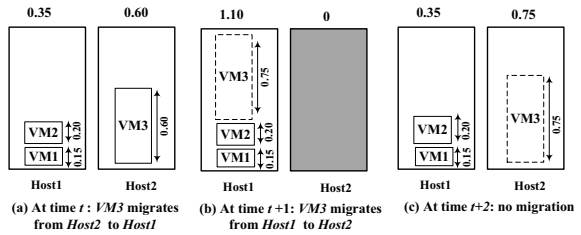

Fig. 1. Example 1

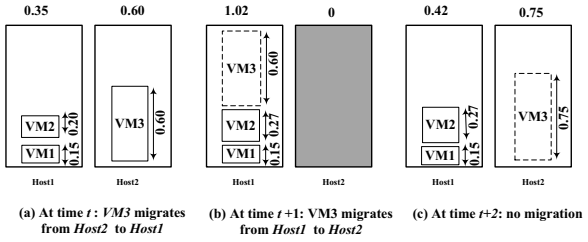

Fig. 2. Example 2

$t+1$, the requested CPU utilization by $VM3$ is increased from 0.60 to 0.75 (Figure 1(b)). As $Host1$ does not have sufficient free capacity that is requested by $VM3$, $Host1$ is overloaded and some SLA violations happen. So $VM3$ migrates to $Host2$ in Figure 1(c) for avoiding further SLA violations. Therefore, a VM consolidation method can avoid the unnecessary migrations and reduce SLA violations if it predicts the resource requirements of a VM before migration.

Another example is shown in Figure 2. At the current time t, $VM3$ is migrated to $Host1$ because it has sufficient capacity for assigning $VM3$ (Figure 2(a)). In Figure 2(b), $Host2$ is switched to the sleep mode, and only $Host1$ is active. As the requested CPU utilization by $VM2$ is increased and hot spots are created, $VM3$ migrates to the previous destination host Figure 2(c) for avoiding SLA violations. So that the number of migrations and SLA violations can be reduced if a VM consolidation method assigns a VM to a host based on the future resource utilization of the host.

IV. System Model

The proposed system architecture is depicted in Figure 3. The virtualized data center consists of M heterogeneous hosts which accommodate N VMs through VMM. Each host is characterized by different resource such as CPU, memory and network bandwidth. At any given time, a cloud data center usually serves many simultaneous users. Users submit their requests for the provisioning of VMs. The length of each request is specified by millions of instructions (MI) and the CPU performance is defined in Million Instructions Per Second (MIPS). The architecture comprises two agents: (1) a Global Agent (GA) resides in a master node, (2) fully distributed Local Agents (LAs) in hosts. The task sequence of each iteration of the proposed dynamic VM consolidation is described as follows:

1) Each LA monitors the resource utilization of hosts and predicts the short-term CPU utilization of a host.
2) The GA collects the utilization data from the LAs and builds a migration plan by using the proposed UP-BFD algorithm, which is described in the next section.
3) The GA sends migration commands to VMMs for performing the VM consolidation task. The commands determine which VMs should be migrated to which hosts.
4) VMMs perform the actual migration of VMs after receiving the commands from the GA.

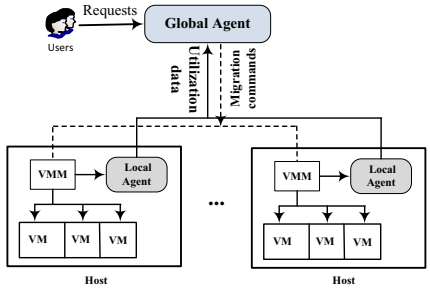

Fig. 3. The system model

V. Utilization Prediction-aware Best Fit Decreasing Algorithm

Initially, VMs are allocated to hosts using the BFD algorithm. The BFD first sorts all VMs based on requested resource utilization in decreasing order. Then it assigns each VM to a host with the smallest free capacity. As the resource utilizations of VMs continue to vary over time due to dynamic workloads, the initial VM placement needs to be augmented with a VM consolidation algorithm that can be applied periodically. For this purpose, we present a dynamic VM consolidation approach called Utilization Prediction-aware Best Fit Decreasing (UP-BFD) that optimizes the VM placement according to the current and future resource requirements. The pseudocode of the UP-BFD algorithm is given in Algorithm 1. For the sake of clarity, the concepts used in the proposed algorithm and their notations are tabulated at Table I.

The UP-BFD algorithm migrates VMs from least-loaded hosts (cold spots) to most-loaded hosts (hot spots) in order to reduce the energy consumption of data centers by releasing cold spots. If all VMs from the coldest spot can not migrate to other hosts, none of them are migrated. Therefore, VM migrations that do not lead to a release of a cold spot will not be carried out and we can thereby eliminate needless VM migrations. Since assigning a VM to a host requires a certain amount of recourses on the host, an efficient VM consolidation algorithm should be dimension aware. It means that the proportionality of resource usage (CPU and memory) should be considered in each host while deciding on VM placement.

TABLE I
SUMMARY OF CONCEPTS AND THEIR NOTATIONS

L_h	load of the host h
L_v	load of the VM v
$C_{CPU}(h)$	total CPU capacity of the host h
$C_{CPU}(v)$	total CPU capacity of the VM V
$C_{MEM}(h)$	total memory capacity of the host h
$C_{MEM}(v)$	total memory capacity of the VM v
$U_{CPU}(h)$	used CPU capacity of the host h
$U_{CPU}(v)$	used CPU capacity of the VM v
$U_{MEM}(h)$	used memory capacity of the host h
$U_{MEM}(v)$	used memory capacity of the VM v
$R_{CPU}(h)$	CPU load of the host h
$R_{CPU}(v)$	CPU load of the VM v
$R_{MEM}(h)$	memory load of the host h
$R_{MEM}(v)$	memory load of the VM v
$PU_{CPU}(h)$	predicted used CPU capacity of the host h
$PU_{CPU}(v)$	predicted used CPU capacity of the VM v
T	CPU utilization threshold value
H_{over}	set of overloaded hosts
$\hat{H}_{over}$	set of predicted overloaded hosts
M	the migration plan
h_s	the source host for migrated VM allocation
h_d	the destination host for migrated VM allocation
v	the migrated VM
V_m	set of selected VMs on a host for migration

As CPU and memory capacity of a host are the key factors to represent the load level, we assumed two resource dimensions which represent CPU and memory utilization. However, we believe it is straightforward to extend our approach to cover other resources such as disk storage. Therefore, the host load is defined as the total resource utilization of each dimension as

$$L_h = R_{CPU}(h) + R_{MEM}(h)$$

$$R_{CPU}(h) = \frac{U_{CPU}(h)}{C_{CPU}(h)}$$

$$R_{MEM}(h) = \frac{U_{MEM}(h)}{C_{MEM}(h)}$$

where $R_{CPU}(h)$ is defined as the CPU capacity used by VMs already allocated on host h ($U_{CPU}(h)$) divided by the total CPU capacity ($C_{CPU}(h)$). In the similar way, $R_{MEM}(h)$ divides the used memory capacity ($U_{MEM}(h)$) by the total memory capacity of host h ($C_{MEM}(h)$).

The UP-BFD algorithm decides which VMs should be migrated from a host to other hosts. It first aims to migrate a VM that has a high load among all VMs on a host. Our idea is based on the fact that bigger VMs are more difficult to insert into the other hosts. The load of the VM v, L_v is defined as

$$L_v = R_{CPU}(v) + R_{MEM}(v)$$

$$R_{CPU}(v) = \frac{U_{CPU}(v)}{C_{CPU}(v)}$$

$$R_{MEM}(v) = \frac{U_{MEM}(v)}{C_{MEM}(v)}$$

where $R_{CPU}(v)$, $R_{MEM}(v)$ are the CPU and memory utilization of the VM v, respectively. $U_{CPU}(v)$ and $U_{MEM}(v)$ show the requested CPU and memory utilization by the VM v, and $C_{CPU}(v)$ and $C_{MEM}(v)$ represent total CPU and memory consumption by the VM.

To find a suitable destination host for allocating a migrated VM, UP-BFD applies two constraints for avoiding SLA violations and needless migrations. The first constraint allows a VM v to allocate the host h_d if it has sufficient resources for allocating the VM. As the utilization of some resource is close to 100% that creates a risk of SLA violation, we consider a threshold value T to limit the amount of CPU resource requested by VMs on a host. We do not consider memory utilization in this context because it is possible that the amount of memory allocated to VMs exceeds the physical memory. Therefore, the first constraint is given as

$$U_{CPU}(h_d) + U_{CPU}(v) \leq T \times C_{CPU}(h_d) \quad (1)$$

The second constraint ensures that the destination host h_d does not become overloaded shortly after allocation of the VM v. For this purpose, UP-BFD considers the future CPU demands by the host and VM. It uses a K-nearest neighbor regression based prediction model [12] to forecast CPU utilization by taking a local average of the training data set. The locality is defined in terms of the k samples nearest to the new sample. We collect the historical CPU usage data set, which each sample of data set consists of one input variable and one output variable. The input variable represents the used CPU capacity at the current time instance and the output variable indicates the CPU utilization at the next time instance. Since the output variable is continuous, the regression analysis is applicable. In addition, UP-BFD should be focused on short-term load prediction due to the dynamic workload. The following second capacity constraint is proposed to assign the VM v to the host h:

$$PU_{CPU}(h_d) + PU_{CPU}(v) \leq T \times C_{CPU}(h_d) \quad (2)$$

where $PU_{CPU}(h_d)$ is the predicted CPU utilization of host h_d and $PU_{CPU}(v)$ is the predicted CPU utilization of VM v. The aggregated predicted CPU utilization of the VM and host is bounded by a specified upper threshold T of the total host capacity. For example, if T= 0.5, the total predicted utilization of host h_d and VM v does not exceed more than 50% of the total host utilization. Based on above two constraints, we select a destination host that has enough available capacity at the current time and the near future for allocating the VM.

In addition, the UP-BFD algorithm aims to migrate some VMs from the overloaded and predicated overloaded hosts in order to reduce the level of SLA violations. If the current CPU utilization exceeds host capacity, the host is considered as a member of overloaded host set (H_{over}). Therefore, some VMs from an overloaded host should migrate to other hosts to reduce the number of SLA violations. Moreover, UP-BFD predicts when a host becomes overloaded based on the predicted CPU utilization of the host. If the predicted utilization value is larger than the available CPU capacity, the host is considered as a member of predicted overloaded hosts ($\hat{H}_{over}$). Therefore, some VMs should migrate from predicated overloaded hosts to guarantee that SLAs are not violated.

The pseudocode in Algorithm 1 consists of two phases: (1) migrating all VMs from a least-loaded host and switching it into the sleep mode to save energy,(2) migrating some VMs from the overloaded and predicted overloaded hosts to reduce SLA violations. At the first phase (lines 2-21), the algorithm sorts hosts in decreasing order based on their load levels and saved the result in the list H (line 2). UP-BFD considers the last host of the list H (least-loaded host) as a source host h_s (line 3) and migrates all VMs in order to release h_s. To select which VMs first migrate from h_s to other hosts, the algorithm sorts all VMs on host h_s in decreasing order based on their load and stores the result in the list V_m (line 4). To find an appropriate destination host for reallocating the migrated VMs, UP-BFD starts from the first host (with the highest load) of list H. If it is not possible, the second host will be selected and so on (lines 5-7). UP-BFD selects a destination host h_d that has the required capacity for allocating the VM at the moment and near future (line 8). Finally, the new VM placement is added to a migration plan M as a member (line 9). The migration plan is a set of 3-tuple $(h_s; v; h_d)$, where the source host h_s, the VM to be migrated v, and the destination host h_d. The CPU used capacity of source and destination hosts are updated to reflect the impact of the migration (line 10). The $success$ variable is defined to check whether all VMs from the source host are migrated or not. Either all VMs from the source host are migrated, if one of them fails non of them are migrated. Therefor, the algorithm removes all tuples in the migration plan and recovers the CPU capacity of the source and destination hosts if the value of $success$ is false (lines 16-18). Otherwise, the source host is switched to the sleep mode when all of its VMs migrate from it, that is, when the node no longer hosts any VMs (line 20).

At the second phase (lines 22-40), the algorithm looks through the list of overloaded and predicted overloaded hosts. It sorts all VMs of a host in decreasing order based on the load level and starts to migrate some VMs that requested maximum capacity (lines 23-24). The VM v migrates to other host when the current CPU utilization or predicted CPU utilization is exceeds the total host utilization(line 25). Then, the algorithm finds the appropriate destination host for reallocating the migrated VM v based on two proposed constraints (line 27). Finally, the new VM placement is added to a migration plan M as a member (line 28). A dormant host is switched to on if the algorithm cannot find a host among active hosts to allocate the VM (lines 33-35). The output of the UP-BFD algorithm is a migration plan and the GA sends commands to VMMs (Figure 3) based on the migration plan.

Algorithm 1 Utilization Prediction-aware Best Fit Decreasing (UP-BFD)

1: $M = \emptyset$
2: $H \leftarrow$ sort all hosts in descending load L_h
3: $h_s \leftarrow$ last host in H
4: $V_m \leftarrow$ sort VMs on host h_s in descending load L_v
5: **for** $v \in V_m$ **do**
6: $success = false$
7: **for** $h_d \in H - h_s$ **do**
8: **if** $(U_{CPU}(h_d) + U_{CPU}(v) \leq T \times C_{CPU}(h_d))$ & $(PU_{CPU}(h_d) + PU_{CPU}(v) \leq T \times C_{CPU}(h_d))$ **then**
9: $M = M \cup \{(h_s, v, h_d)\}$
10: Update $U_{CPU}(h_s)$ and $U_{CPU}(h_d)$
11: $success = true$
12: break;
13: **end if**
14: **end for**
15: **end for**
16: **if** $success = false$ **then**
17: $M = \emptyset$
18: Recover $U_{CPU}(h_s)$ and $U_{CPU}(h_d)$
19: **else**
20: Switch h_s to the sleep mode
21: **end if**
22: **for** $h_s \in [H_{over}, \hat{H}_{over}]$ **do**
23: $V_m \leftarrow$ sort VMs on host h_s in descending load L_v
24: **for** $v \in V_m$ **do**
25: **if** $((U_{CPU}(h_s) \geq C_{CPU}(h_s)) || (PU_{CPU}(h_s) \geq C_{CPU}(h_s))$ **then**
26: **for** $h_d \in H - [H_{over}, \hat{H}_{over}]$ **do**
27: **if** $(U_{CPU}(h_d) + U_{CPU}(v) \leq T \times C_{CPU}(h_d))$ & $(PU_{CPU}(h_d) + PU_{CPU}(v) \leq T \times C_{CPU}(h_d))$ **then**
28: $M = M \cup \{(h_s, v, h_d)\}$
29: Update $U_{CPU}(h_s)$ and $U_{CPU}(h_d)$
30: break;
31: **end if**
32: **end for**
33: **if** $((U_{CPU}(h) \geq C_{CPU}(h)) || (PU_{CPU}(h) \geq C_{CPU}(h))$ **then**
34: Switch on a dormant host
35: **end if**
36: **else**
37: break;
38: **end if**
39: **end for**
40: **end for**

VI. Performance Evaluation

A. Simulation Setup

To evaluate the efficiency of our proposed dynamic VM consolidation approach, we implement various VM consolidation techniques using the CloudSim toolkit [13]. We simulated a data center consist of 800 heterogeneous hosts and selected two server configurations: HP ProLiant ML110 G4 (Intel Xeon 3040, 2 cores $\times$1860 MHz, 4 GB), and HP ProLiant ML110 G5 (Intel Xeon 3075, 2 cores $\times$ 2660 MHz, 4 GB). Each server is modeled to have 1 GB/s network bandwidth. The number of VMs depends on the type of workload. Two types of workloads including random and real are used. In the random workload, the users submit requests for the provisioning of 800 heterogeneous VMs where each VM runs an application

with a variable utilization of CPU generated with a uniform distribution. In the real workload, the number of VMs on each day of March 2011 load trace is specified in Table II. Real workload data is provided as a part of the CoMon project, a monitoring infrastructure for PlanetLab [14]. In this project, the CPU usage data is collected every five minutes from more than a thousand VMs and is stored in different files. The VMs are allocated on servers that are located at more than 500 places around the world. In fact, the workload is representative of an IaaS cloud environment such as Amazon EC2, where several independent users create and manage VMs with the only exception that all the VMs are single-core, which is explained by the fact that the workload data used for the simulations come from single-core VMs. For the same reason the amount of RAM is divided by the number of cores for each VM type: High-CPU Medium Instance (2500 MIPS, 0.85 GB); Extra Large Instance (2000 MIPS, 3.75 GB); Small Instance (1000 MIPS, 1.7 GB); and Micro Instance (500 MIPS, 613 MB).

TABLE II
THE NUMBER OF VMS IN THE REAL WORKLOAD

Date	Number of VMs	Mean%	St.dev.%
3 March	1052	12.31	17.09
6 March	898	11.44	16.83
9 March	1061	10.70	15.57
22 March	1516	9.26	12.78
25 March	1078	10.56	14.14

B. Evaluation Metrics

The main goals of our approach are to: i) guarantee that SLAs are not violated; ii) minimize the number of physical servers used; iii) minimize the number of migrations. Therefore, the performance of proposed methods is assessed through the following metrics:

- SLA Violations: a workload independent metric (SLAV) is proposed in [3] that can be used to evaluate the SLA delivered by any VM deployed in an IaaS. SLA violations can measure by the SLA violations due to over-utilization (SLAVO) and SLA violations due to migration (SLAVM). Both SLAVO and SLAVM metrics independently and with equal importance characterize the level of SLA violations by the infrastructure. Therefore, a combined metric (SLAV) can describe both performance degradation due to host overloading and due to VM migrations as

$$SLAV = SLAVO \times SLAVM$$

SLAVO indicates the percentage of time, during which active hosts have experienced the CPU utilization of 100% as

$$SLAVO = \frac{1}{M}\sum_{i=1}^{M}\frac{T_{s_i}}{T_{a_i}}$$

where M is the number of PMs; T_{s_i} is the total time that the PM i has experienced the utilization of 100% leading to an SLA violation. T_{a_i} is the total of the PM i being the active state. SLAVM shows the overall performance degradation by VMs due to migrations as

$$SLAVM = \frac{1}{N}\sum_{j=1}^{N}\frac{C_{d_j}}{C_{r_j}}$$

where N is the number of VMs; C_{d_j} is the estimate of the performance degradation of the VM j caused by migrations; C_{r_j} is the total CPU capacity requested by the VM j during its lifetime. In our experiments, we estimate C_{d_j} as 10% of the CPU utilization in MIPS during all migrations of the VM j.
- Energy consumption: we consider the total energy consumption by the physical resources of a data center caused by application workloads. The energy consumption of PMs depends on the utilization of a CPU, memory, disk and network card. Most studies [3], [15] show that CPU consumes more power than other devices such as memory, disk storage and network interface. Therefore, the resource utilization of a PM is usually represented by its CPU utilization. Here the energy consumption is measured based on real data on power consumption provided by the results of the SPECpower benchmark[1] instead of using an analytical model of server power consumption. Table III illustrates the amount of energy consumption of two types of HP G4 and G5 servers in different load levels.
- Number of VM Migrations: live VM migration is a costly operation that involves some amount of CPU processing on a source PM, the link bandwidth between the source and destination PMs, downtime of the services on a migrating VM and total migration time [4]. Therefore, our objective is to minimize the number of migrations.

C. Comparison benchmarks

In order to investigate the effectiveness of the utilization prediction on VM consolidation performance, we implement four following heuristic algorithms:

- Modified Best Fit Decreasing (MBFD): in order to applicable comparison with proposed methods, we adapted BFD to our model. We assume a threshold value to bound the CPU utilization of a host for allocating VMs. This algorithm assigns a VM to a host with the smallest free capacity if the total used CPU capacity of the VM and host does not exceed the threshold of the total CPU utilization:

$$U_{CPU}(h_d) + U_{CPU}(v) \leq T \times C_{CPU}(h_d)$$

- Modified First Fit Decreasing (MFFD): is a modified version of FFD algorithm. The MFFD allocates a VM to the first host that the total used CPU capacity of the VM and host does not exceed the threshold of the total CPU utilization.

$$U_{CPU}(h_d) + U_{CPU}(v) \leq T \times C_{CPU}(h_d)$$

[1] $http://www.spec.org/power\_ssj2008/$

TABLE III
THE ENERGY CONSUMPTION AT DIFFERENT LOAD LEVELS IN WATTS

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

- Host Utilization Prediction-aware Best Fit Decreasing (HUP-BFD): uses the prediction model and threshold value T to limit the host utilization. The VM v can allocate to the host h_d if the aggregated predicted host utilization and the requested CPU utilization of the VM does not exceed of the threshold of the total CPU utilization:

$$PU_{CPU}(h_d) + U_{CPU}(v) \leq T \times C_{h_d}$$

- VM Utilization Prediction-aware Best Fit Decreasing (VMUP-BFD): this algorithms considers the future requested utilization of a VM that is predicted by using K-nearest neighbor regression. The VM v can be assigned to the destination host h_d if the aggregated used CPU capacity of the host and the predicted requested usage by the VM does not exceed of the threshold of the total CPU utilization:

$$U_{CPU}(h_d) + PU_{CPU}(v) \leq T \times C_{h_d}$$

D. Experimental Results

We conducted several experiments to evaluate the performance of our approach. In the first experiments, we generated a random workload based on a uniform distribution. Assuming a high value for the CPU threshold leads to high performance degradation and number of migrations and, a low threshold value can increase energy consumption. So finding a suitable threshold value is essential for considering the tradeoff between the evaluation metrics in the data center. For this purpose, we obtained results on various threshold values between 50% to 100% to determine the impact of threshold value on the performance metrics. Figure 4(a) presents the SLA violation levels caused by UP-BFD, HUP-BFD, VMUP-BFD, MFFD, and MBFD methods for different threshold values. The UP-BFD method can reduce the percentage of SLA violation rate more efficiently than the other methods. The results can be explained by the fact that UP-BFD allocates a VM to a host that does not become overloaded in the near future besides the current time. In addition, UP-BFD tries to migrate VMs from the overloaded and predicted overloaded hosts in order to avoid further SLA violations. Figure 4(b) shows our proposed VM consolidation approach can reduce energy consumption by up to 28.5% with desirable system performance. This is because, the proposed method minimize the number of active hosts by packing VMs into the most-loaded hosts. Figure 4(c) depicts the total number of VM migrations during the VM consolidation in the random workload. UP-BFD outperforms the benchmark algorithms due to the predictions of utilization, and therefore decreased the number of VM migrations.

In the second experiment, we run the simulation for the real workload with large number of VMs. During the simulation, each VM is randomly assigned a workload trace from one of the VMs from the corresponding day. The results in Figure 5 show the average value of each performance metric on five workloads. Figure 5(a) shows that UP-BFD leads to significantly less SLA violations than the other four benchmark algorithms. This is due to the fact that UP-BFD prevents SLA violations by using a prediction of host and VM utilization and ensures that the destination host does not become overloaded when a VM migrates on it. Figure 5(b) depicts UP-BFD brought higher energy savings in comparison to the other approaches in the real workload. This is because, UP-BFD tries to release least-loaded hosts by packing VMs into a few number of hosts. As observed from the results, UP-BFD has the minimum number of migrations compared with the other benchmark methods. This is due to the fact it allocates a VM to a host according to the future resource utilization. Moreover, the obtained results in Figure 4 and 5 show the tradeoff between energy consumption and SLA violations. The number of migrations and SLA violations are increased if the threshold is set to a high value. However, UP-BFD achieves more energy saving if the threshold value is close to 100%,

E. Prediction Performance

Our proposed VM consolidation approach exploits the prediction model to proactively packing VMs into few hosts. Therefore, we assess the prediction model accuracy by using Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE). To evaluate the quality of our prediction technique, we used five different data sets. Each data set collects the CPU utilization of a host by running real workload traces. Thus a one-step prediction is equivalent to predict the host utilization in the next five minutes. The data sets are divided into a training data set and validation data set. The training data set used to determine the number of neighbors (k) in K-NNR and the validation data set is then used assess the accuracy of the prediction model. RMSE and MAPE are 0.24 and 0.26, respectively. The small values of RMSE and MAPE can prove the K-NNR provides an accurate prediction of utilization at the next time step.

VII. CONCLUSION

In this paper, we presented a dynamic Virtual Machine (VM) consolidation approach which eliminates unnecessary VM migrations and reduces SLA violations using a utilization prediction model. This approach aims to migrate some VMs from physical hosts that are currently overloaded or predicted to become overloaded in the near future. Moreover, it assigns a VM to a host according to current and future resource requirements. Compared with other VM consolidation algorithms, our

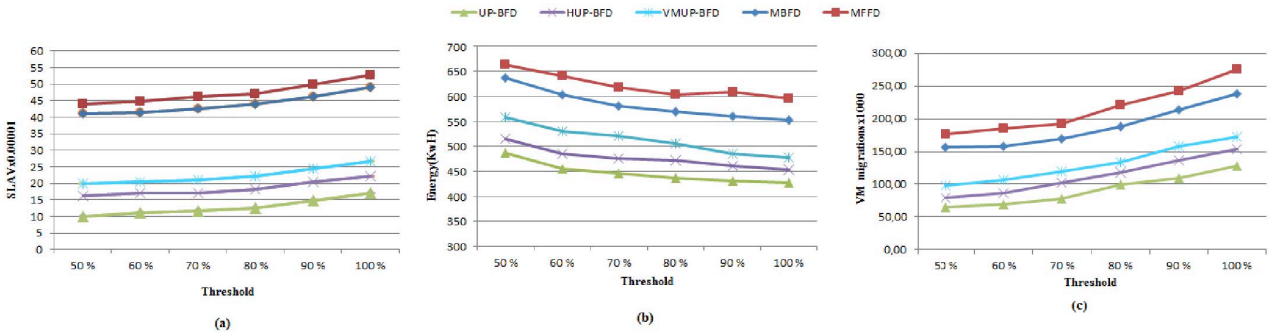

Fig. 4. The SLAV metric, energy consumption and number of VM migrations by UP-BFD and benchmark methods in the random workload

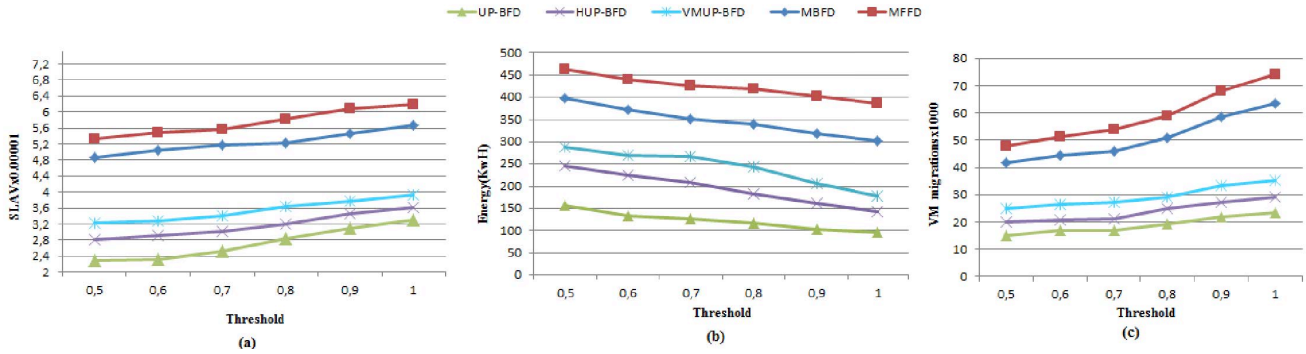

Fig. 5. The SLAV metric, energy consumption and number of VM migrations by UP-BFD and benchmark methods in the real workload

approach not only reduced the energy consumption, but also minimized the number of SLA violations and VM migrations.

REFERENCES

[1] M. Rosenblum and T. Garfinkel, "Virtual machine monitors: current technology and future trends," *Computer*, vol. 38, no. 5, pp. 39–47, May 2005.

[2] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation*, ser. NSDI'05, vol. 2, 2005, pp. 273–286.

[3] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[4] 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, pp. 212–231, 2011.

[5] F. Farahnakian, P. Liljeberg, and J. Plosila, "LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers," in *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on*, 2013, pp. 357–364.

[6] T. Cover and P. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on*, vol. 13, no. 1, pp. 21–27, 1967.

[7] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, May 2007, pp. 119–128.

[8] F. Farahnakian, A. Ashraf, T. Pahikkala, P. Liljeberg, J. Plosila, I. Porres, and H. Tenhunen, "Using ant colony system to consolidate vms for green cloud computing," *Services Computing, IEEE Transactions on*, vol. 8, no. 2, pp. 187–198, March 2015.

[9] A. Verma, P. Ahuja, and A. Neogi, "Pmapper: Power and migration cost aware application placement in virtualized systems," in *9th ACM/IFIP/USENIX International Conference on Middleware*, 2008, pp. 243–264.

[10] T. H. Nguyen, M. D. Francesco, and A. Ylä − Jääski, "A virtual machine placement algorithm for balanced resource utilization in cloud data centers," *The 7th IEEE International Conference on Cloud Computing (IEEE CLOUD)*, p. 474481, 2014.

[11] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong, "Enacloud: An energy-saving application live placement approach for cloud computing environments," *2013 IEEE Sixth International Conference on Cloud Computing*, vol. 0, pp. 17–24, 2009.

[12] F. Farahnakian, T. Pahikkala, P. Liljeberg, and J. Plosila, "Energy aware consolidation algorithm based on k-nearest neighbor regression for cloud data centers," in *Utility and Cloud Computing (UCC), 2013 IEEE/ACM 6th International Conference on*, Dec 2013, pp. 256–259.

[13] 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.

[14] K. Park and V. Pai, "CoMon: a mostly-scalable monitoring system for PlanetLab," *ACM SIGOPS Operating Systems Review*, vol. 40, pp. 65–74, 2006.

[15] D. Kusic, J. Kephart, J. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," in *Autonomic Computing, 2008. ICAC '08. International Conference on*, June 2008, pp. 3–12.