This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2617374, IEEE Transactions on Cloud Computing

Energy-aware VM Consolidation in Cloud Data Centers Using Utilization Prediction Model

Fahimeh Farahnakian, Tapio Pahikkala, Pasi Liljeberg, Juha Plosila, Nguyen Trung Hieu, and Hannu Tenhunen

Abstract—Virtual Machine (VM) consolidation provides a promising approach to save energy and improve resource utilization in data centers. Many heuristic algorithms have been proposed to tackle the VM consolidation as a vector bin-packing problem. However, the existing algorithms have focused mostly on the number of active Physical Machines (PMs) minimization according to their current resource requirements and neglected the future resource demands. Therefore, they generate unnecessary VM migrations and increase the rate of Service Level Agreement (SLA) violations in data centers. To address this problem, we propose a VM consolidation approach that takes into account both the current and future utilization of resources. Our approach uses a regression-based model to approximate the future CPU and memory utilization of VMs and PMs. We investigate the effectiveness of virtual and physical resource utilization prediction in VM consolidation performance using Google cluster and PlanetLab real workload traces. The experimental results show, our approach provides substantial improvement over other heuristic and meta-heuristic algorithms in reducing the energy consumption, the number of VM migrations and the number of SLA violations.

Index Terms—VM consolidation, linear regression, k-nearest neighbor regression, energy-efficiency, SLA, green computing.

1 Introduction

Energy-related costs and environmental impacts of data centers have become major concerns and, research communities are being challenged to find efficient energy-aware resource management strategies. VM consolidation is an efficient way towards energy conservation in data centers. It leverages virtualization technology such as Xen [1] and VmWare [2] to improve resource utilization by supporting multiple enterprise applications using fewer computing resources. Virtualization allows multiple Virtual Machines (VMs) to share resources on a Physical Machine (PM) through VM monitor or hypervisor. In addition, virtualization provides the ability to transfer a VM between PMs using live migration [3]. VM consolidation through live migration has a major impact on energy efficiency by packing VMs into the minimum number of PMs and switching the idle PMs to a power-saving mode. Migrating a VM can be advantageous either when a PM is highly under-loaded, or when it is over-loaded. However, unreliable characterization of over-loaded and under-loaded PMs may cause unnecessary live migration, thus impacting on the performance of applications running in a VM during a migration. Voorsluys et al. [4] have performed an experimental study to investigate this impact and to find a way to model it. Hence, how to perform VM consolidation operation with the minimum number of migrations is a challenging task.

Most of the recent works formulate VM consolidation as a bin-packing problem where the PMs are conceived as bins and VMs as items. The classical bin-packing algorithms should be modified to apply in the VM consolidation problem for three main reasons. First, PMs are characterized with multi-dimensional resources such as CPU, memory, and network bandwidth. For instance, VM consolidation becomes a bi-dimensional bin-packing problem with considering the CPU and memory utilization of PMs. Therefore, a VM consolidation algorithm should be concerned with different resource dimensions to balance the usage of resources across multiple dimensions. Second, VM consolidation can be seen as a bin-packing problem with different bin sizes, i.e., heterogeneous PMs, unlike the classical bin-packing problem where bin capacities are equal. Third, the classical bin-packing algorithms only rely on minimizing the number of bins (single-objective), although we should take into account multi-objective such as the number of migrations and SLA for designing an efficient VM consolidation algorithm. Therefore, VM consolidation is a NP-hard problem, and the optimization algorithms can find a near optimal solution. For this reason, we presented a dynamic VM consolidation approach that uses a highly adaptive online optimization bio-inspired algorithm called Ant Colony System (ACS) in our previous work [5]. The proposed ACS-based VM Consolidation (ACS-VMC) approach uses artificial ants to consolidate VMs into a reduced number of active PMs according to the current resource requirements. These ants work in parallel to build VM migration plans based on a specified objective function. ACS is especially attractive for VM consolidation due to find a solution close to the optimal solutions, and the ease of parallelization.

In this paper, we address the VM consolidation problem with presenting a Utilization Prediction-aware VM Consolidation (UP-VMC) approach. Our main contributions are as follows:

- *F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, and H. Tenhunen are with the Department of Information Technology, University of Turku, Turku, Finland.*
  E-mail: fahfar@utu.fi, aatapa@utu.fi, pakrli@utu.fi, juplos@utu.fi, hanten@utu.fi
- *T. H. Nguyen is with Department of Computer Science, Aalto University, Finland.*
  E-mail: nguyen.hieu@aalto.fi

- UP-VMC formulates VM consolidation as a bi-dimensional vector packing problem. UP-VMC goes beyond the existing works which only consider CPU utilization by also considering memory. Combining both memory and CPU utilization, UP-VMC can better identify causes of SLA violations and consequently prevent them from happening.
- In contrast to the existing VM consolidation methods which mostly rely on the current resource utilization of PMs, UP-VMC considers both current and future resource utilization. In order to approximate the future utilization, we propose two regression-based prediction models: linear and k-nearest neighbor.
- We implemented and evaluated UP-VMC on a simulated large-scale data center using the real Google and PlanetLab workloads. We experimentally show the added value of employing the utilization prediction model for VM consolidation by comparison with Sercon and a modified version of FF and BF. We evaluate the benefits of a greedy-based VM consolidation algorithm by comparing with ACS-VMC. We also conduct a comprehensive study of the effectiveness of utilization prediction-aware approaches on the VM consolidation problem. For this purpose, we consider three different scenarios: when UP-VMC uses the prediction model in order to predict (1) resource utilization of VMs; (2) resource utilization of PMs; and (3) resource utilization of both VMs and PMs.
- We investigate how four well-known VM selection methods can affect on the performance of UP-VMC in terms of the energy consumption, the number of SLA violations and the number of migrations. The simulation results show that the performance of VM consolidation is increased when the VM allocation algorithm selects a VM that requires the minimum time for migration to another PM.
- 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%.

The rest of this paper is organized as follows: in Section 2 surveys some literature regarding to dynamic consolidation in cloud data centers. Section 3 presents the proposed system architecture and the VM consolidation algorithm. Section 4 shows the implementation issue of our algorithm. Finally, we give the experimental results and conclusion in Section 5 and 6.

2 Related Work

In recent years, there have been major significant research in data center energy efficiency. The most of research focus on VM consolidation methods as an emerging solution to save energy in cloud data centers. The main idea in the existing VM consolidation approaches is to use live migration to consolidate VMs periodically [6], [7], [8]. These approaches reduce the power consumption by packing the existing VMs into fewer PMs and switching the idle PMs into a power saving mode. Sandpiper [9] is a system that is enabled to detect over-utilized PMs and creates a new mapping of physical resources to virtual resources. To detect overloaded PMs, Sandpiper collects VM and PM usage statistics, constructs profiles of resource usage and then uses the prediction techniques. Bobroff et al. [10] presents a dynamic server migration and consolidation algorithm which uses time series forecasting techniques and bin packing heuristic to minimize the number of PMs required to support a workload. Their algorithm does not take into account the number of migrations required to a new placement.

A VM consolidation algorithm should incorporate both performance and power considerations into its decision making on new VM placement. For this purpose, Sercon [11] considers a threshold value to prevent CPU's PM from reaching 100% utilization that leads to performance degradation. Therefore, it tries to keep the total usage of a PM 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 PM. Therefore, the threshold value should be tuned for each workload type and level to allow the consolidation task to perform efficiently. Beloglazov and Buyya [12] proposed adaptive upper and lower thresholds based on the statistical analysis of the historical data. A VM consolidation method can minimize the number of SLA violations by forecasting future resource utilization. Our previous works in [13], [14] use a prediction model to forecast PMs' utilization. If the predicted usage exceeds of the available capacity of a PM, some VMs are reallocated to other PMs for avoiding a SLA violation. Moreover, a resource usage prediction algorithm is presented in [15] to predict the system usage by interpolating what follows after the identified patterns from the historical data. This algorithm uses a set of historical data to identify similar usage patterns to a current window of records that occurred in the past.

In addition, a lot of research has been done on designing heuristic algorithms for solving the VM consolidation as a bin packing problem. This problem is known to be NP-hard, and there are many heuristic algorithms to solve it. In [16] the authors survey the existing greedy methods in order to tackle one-dimensional bin-packing problem. Among the most popular heuristics, the 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 it 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). However, the classical bin-packing algorithms cannot be used directly for VM consolidation and they should be modified for applying to the consolidation problem. Therefore, pMapper [17] applies a modified version of the FFD heuristic to perform server consolidation. It presents a power and migration cost aware application placement controller in heterogeneous server clusters that support virtualization with live VM migration. In addition, we proposed a modified BFD algorithm in [18] to predict only CPU resource utilization based on K-nearest neighbor regression.

In [12], 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. Similarly in [19], a framework called EnaCloud is pre-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2617374, IEEE Transactions on Cloud Computing

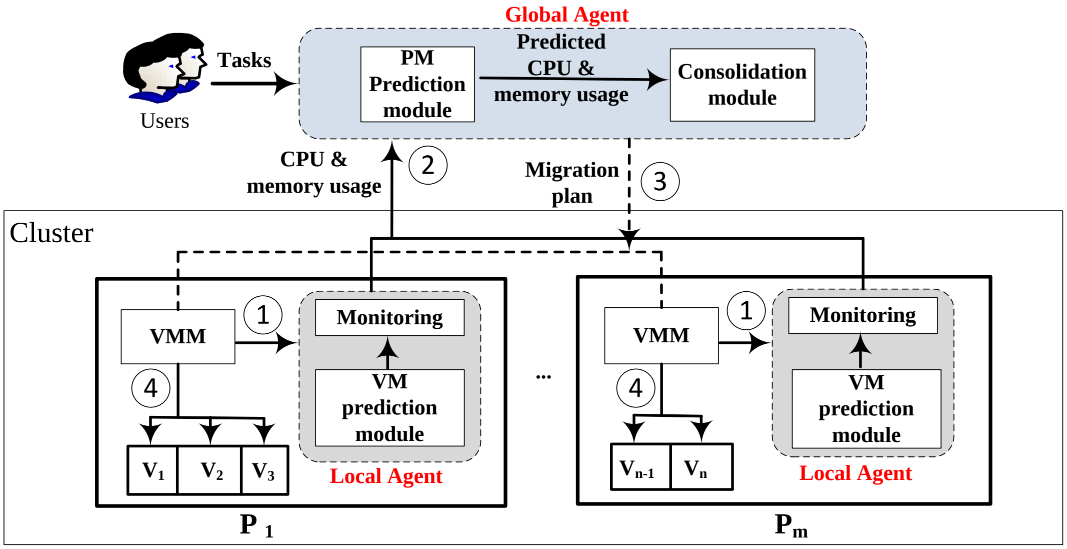


Fig. 1. The system architecture.

sented 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 other important performance metrics. Another work [20] models the resource allocation as a multi-dimensional bin packing problem and provides simulation results for many greedy algorithms (e.g., FF, BF, Permutation Pack (PP), Choose Pack (CP)). Thereby, the objective of these algorithms is only to maximize the minimum yield over all services.

In our earlier work [8], an ant colony system based approach is proposed for solving the VM consolidation problem as one-dimensional bin packing problem. The work in [8] is then refined in [5]. In that context, we proposed a dynamic VM consolidation approach that uses a highly adaptive online optimization bio-inspired algorithm called Ant Colony System (ACS) to optimize VM placement. The proposed ACS-based VM Consolidation (ACS-VMC) approach uses artificial ants to consolidate VMs into a reduced number of active PMs according to the current resource requirements. These ants work in parallel to build VM migration plans based on a specified objective function. With respect to [5], [8], we extend the ACS-based VM consolidation algorithm that models VM consolidation as a multi-dimensional bin packing problem. However, this algorithm is not scalable in the large scale data center due to the cost of increased execution time.

To address this problem, we present a greedy-based VM consolidation approach, named Utilization Prediction-aware VM Consolidation (UP-VMC) in this paper. UP-VMC formulates the VM consolidation as a multi-objective vector bin packing problem in order to optimize three conflicting objectives simultaneously. The objectives include reducing energy consumption, minimizing the number of VM migrations, and avoiding SLA violations. Generally, UP-VMC performs two essential steps in order to reduce SLA violations and energy consumption: (1) migrating the number of VMs from PMs that are over-loaded currently or become over-loaded in the short term of future and (2) migrating all VMs from the least-loaded PMs and then switch the idle PM to the sleep mode. Moreover, UP-VMC selects a PM to allocate a VM based on the current and future resource utilization of PM and VM in order to avoid unnecessary VM migrations. The future resource utilization is approximated using a regression-based model as our previous works [14], [13] have shown that the regression-based model is more efficient than the robust statistic methods in order to forecast resource utilization. Finally, we evaluate our proposed approach using Google and PlanetLab workloads. We found UP-VMC achieves lower SLA violations and energy consumption compared to the classical bin packing algorithms, Sercon [11] and ACS-VMC algorithm [5].

3 Utilization Prediction Aware VM Consolidation

In this section, we present a Utilization Prediction-aware VM Consolidation (UP-VMC) approach for cloud data centers. We first introduce the proposed system architecture, then we explain the UP-VMC algorithm. Finally, an example of UP-VMC is presented to clarify the algorithm.

3.1 System Architecture

We consider a data center that consists of m heterogeneous PMs, $P = \langle p_1, ..., p_m \rangle$. Each PM is characterized with D type of resources such as CPU, memory, network I/O and storage capacity. In addition, multiple VMs can be allocated to each PM through Virtual Machine Monitor (VMM). In our implementation, the VMs are initially allocated to PMs based on the Best Fit Decreasing (BFD) algorithm, which is one of the best known heuristics for the bin-packing problem. At any given time, users submit their requests for provisioning of n VMs, $V = \langle v_1, ..., v_n \rangle$, which are allocated to the PMs. As the requested utilization of VMs and PMs vary over time, an initial efficient allocation approach needs to be augmented with a VM consolidation algorithm that can be applied periodically. In our proposed approach, the UP-VMC algorithm is applied periodically in order to adapt and optimize the VM placement according to the workload. Figure 1 shows the proposed system architecture that consists of two kind of agents: (1) fully distributed Local Agents (LAs) in PMs, (2) a Global Agent (GA) resides in a master node. The task sequence of these agents is described as follows:

1) Each LA monitors the current resource utilization of all VMs in a PM periodically. Then it approximates the future utilization of all VMs in a PM using a regression-based prediction model (i.e., linear regression or k-nearest neighbor regression).
2) The GA collects the information from the local agents to maintain the overall view of current and future resource utilization of VMs.
3) The GA builds a global best migration plan to optimize VM placement by using the UP-VMC algorithm, which is described in the next section. Then it sends commands to VMMs for performing VM placement. The commands determine which VMs on a source PM should be migrated to which destination PMs.
4) The VMMs perform actual migration of VMs after receiving the commands from the GA.

TABLE 1. Summary of concepts and their notations

m	number of PMs in the data center
n	number of VMs in the data center
P_{active}	set of active PMs
P_{over}	set of over-loaded PMs
$\hat{P}_{over}$	set of predicted over-loaded PMs
V_p	set of VMs running on a PM p
V_m	set of VMs for migration
p_{de}	destination PM
p_{so}	source PM
v	VM in a tuple
$C_{p_{de}}$	total capacity vector of the destination PM p_{de}
$C_{p_{so}}$	total capacity vector of the source PM p_{so}
U_v	used capacity vector of the VM v
$U_{p_{de}}$	used capacity vector of the destination PM p_{de}
$U_{p_{so}}$	used capacity vector of the source PM p_{so}
$PU_{p_{de}}$	predicted used capacity vector of the destination PM p_{de}
PU_v	predicted used capacity vector of the VM v
$Load_v$	load level of the VM v
$Load_p$	load level of the PM p
M	final migration plans
$M1$	migration plans is generated by the first step of UP-VMC
$M2$	migration plans is generated by the second step of UP-VMC

3.2 The proposed Algorithm

Each PM p has a d-dimensional total capacity vector $C_p = \langle C_p^1, C_p^2, ..., C_p^d \rangle$, where C_p^d represents the total d-th resource capacity of PM p. Each dimension corresponds to one type of physical resource (e.g., CPU capacity, memory, network I/O and disk storage). According to the resource dimensions of the real-world workloads, we set $|D| = 2$ and consider a bi-dimensional VM consolidation scheme. We do not take into account the disk size dimension because we assume that network-attached storage (NAS) is used as main storage across the data center. However, if necessary, it is possible to add more dimensions in the total and used capacity vectors. In addition, the used capacity vector of the PM p can be represented as $U_p = \langle U_p^1, U_p^2, ..., U_p^d \rangle$, where U_p^d denotes the used capacity of the resource type d. For instance, the used CPU capacity of a PM is estimated as the sum of the CPU utilization of the three VMs if three VMs are hosted by the same PM. At any given time, the data center usually serves many simultaneous independent users. Users submit their requests for the provisioning of VMs that are located on the PMs. The total capacity vector of the VM v is presented as $C_v = \langle C_v^1, C_v^2, ..., C_v^d \rangle$, where C_v^d indicates the total d-th consumption by the VM v. Moreover, the d-dimensional used capacity vector $U_v = \langle U_v^1, U_v^2, ..., U_v^d \rangle$ determines the used d-th resource of the VM v. As the resource utilization of VMs vary over time due to dynamic workloads, the VM placement is optimized with the proposed UP-VMC algorithm that can be applied periodically. The pseudocode of the UP-VMC algorithm is given as Algorithm 1. For the sake of clarity, the concepts used in the proposed algorithm and their notations are tabulated in Table 1. The UP-VMC algorithm run two steps to formulate the VM consolidation problem.

First step- the UP-VMC algorithm aims to migrate some VMs from the over-loaded and predicated over-loaded PMs. If at least one resource (i.e., CPU or memory) exceeds total capacity, PM is considered as a member of over-loaded PM set (P_{over}). If at least one predicted utilization value is larger than the available resource capacity, PM is considered as a member of predicted over-loaded PMs ($\hat{P}_{over}$). UP-VMC first aims to migrate some VMs from P_{over} and repeats the process until there are no over-loaded PMs left. Then, some VMs migrate from $\hat{P}_{over}$ to guarantee that SLAs are not violated in the near of future. Generally, the goal of this step is to move some VMs from over-loaded and predicted over-loaded PMs for minimizing SLA violations.

In order to predict the resource utilization, we assume two regression-based prediction models: Linear Regression (LR) [21] and K-Nearest Neighbor Regression (K-NNR) [22]. The prediction models estimate the resource utilization of VMs and PMs in terms of CPU and memory. Since the CPU and memory utilization are continuous, the regression-based prediction model is applicable. UP-VMC focuses on short-term load prediction due to the dynamic workload. LR uses the historical resource utilization data and estimates a linear function. The function determines the relationship between the current and future resource utilization as

$$PU_{p_{de}} = \alpha + \beta U_{p_{de}}, \quad (1)$$

where $PU_{p_{de}}$ and $U_{p_{de}}$ are the predicted and current used capacity vector of the PM p, respectively. α and β are regression coefficients specifying the Y-intercept and slope of the line, respectively. The most popular method for estimating these coefficients is the least square method. This method estimates the best-fitting straight line as the one that minimizes the distance between the observed output and the predicted output in the data set by the linear approximation. Therefore, the regression coefficients can be estimated using this method by the following equations:

$$\beta = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

$$\alpha = \bar{y} - \beta \bar{x},$$

where $\bar{x}$ is the mean value of $x_1, ..., x_n$, and $\bar{y}$ is the mean value of $y_1, ..., y_n$.

In a similar way, the resource utilization of a VM is predicted based on a linear function which represents the relationship between the current used capacity vector U_v and the predicted used capacity vector PU_v of the VM v.

$$PU_v = \alpha + \beta U_v. \tag{2}$$

In addition, we also employ K-NNR to predict the future utilization of PM and VM based on a historical utilization data set. In the other word, K-NNR estimates the future utilization by taking a local average of the data set. The locality is defined in terms of the k samples nearest to the new sample. The best value of k (number of neighbors) is obtained by leave-one-out cross-validation which is a value between one and the number of samples. Leave-one-out cross-validation estimates the prediction accuracy by summing of squared residuals for each k value. Then, it selects the best value for k that has the minimum total residual.

UP-VMC selects some VMs to migrate from a over-loaded or predicted over-loaded PM based on a VM allocation policy. For this purpose, we assume three well-known VM selection policies: Minimum Migration Time (MMT), Maximum Load (MaxL) and Minimum Load (MinL). In comparison between the proposed VM selection polices (see Section 5.1), the UP-VMC algorithm provides more energy-efficient solution with fewer SLA violations and number of migrations when it uses MMT as a VM selection policy. MMT selects a VM that requires the minimum time for migration. The migration time is calculated as the amount of memory by the VM divided by the spare network bandwidth available for the PM.

Second step- the UP-VMC algorithm aims to eliminate as most the least-loaded PMs (cold spots) as possible for energy saving purpose in the data center. It migrates all VMs from the cold spots to the most-loaded PMs (hot spots) in order to reduce the energy consumption of data centers by releasing the cold spots. UP-VMC guarantees a destination PM does not become over-loaded in the near future after placing the migrated VM. If the sufficient resources are not available in the first hot spot, then UP-VMC considers the next most-loaded PM and so on, until a match is found for allocating the VM. If all VMs from a cold spot cannot migrate to other PMs, none of them are migrated. Assigning a VM to a PM requires a certain amount of different resources on the PM, so that the algorithm captures the measures of overall resource utilization for multiple resource types. It means that the proportionality of CPU and memory usage are considered in each PM for VM allocation. Therefore, the load of PM p is modeled as the summation of the resource utilization ratio R_p^d in each individual resource $d \in D$ as

$$Load_p = \sum_{d \in \{1,\ldots,|D|\}} R_p^d, \tag{3}$$

where R_p^d is the ratio of its utilized resource U_p^d to its total resource C_p^d as

$$R_p^d = \frac{U_p^d}{C_p^d}. \tag{4}$$

To determine which VMs to migrate from the cold spot, the algorithm orders VMs in decreasing order of their load. Therefore, first a VM is selected for migration that has a high load among all VMs. Our idea is based on the fact that bigger VMs are more difficult to insert into the other PMs. The load level of VM v is defined as

$$Load_v = \sum_{d \in \{1,\ldots,|D|\}} R_v^d, \tag{5}$$

where R_v^d is the ratio of the requested d-th utilization of VM v to the total d-th consumption by VM v.

$$R_v^d = \frac{U_v^d}{C_v^d}. \tag{6}$$

To find a suitable destination PM for allocating a migrating VM, we apply two constraints for avoiding SLA violations and needless migrations. The first constraint allows a VM v to allocate the PM p_{de} if the PM has sufficient resources for allocating the VM at the moment. A risk of SLA violation is created when the utilization of a PM is close to 100%. So we propose an upper threshold value T to limit the amount of requested resource by VMs on a PM and reduce the SLA violations. Therefore, the first capacity constraint is given as

$$U_{p_{de}} + U_v \leq T \times C_{p_{de}}, \tag{7}$$

where $C_{p_{de}}$ is the total capacity vector of the destination PM p_{de}, $U_{p_{de}}$ is the used capacity vector of p_{de}, and likewise U_v is the used capacity vector of the VM v.

The second constraint ensures the destination PM does not become over-loaded in the near of future after VM allocation. For this purpose, the UP-VMC algorithm considers the future resource capacity of both the PM p_{de} and VM v. The second predicted capacity constraint is given as

$$PU_{p_{de}} + PU_v \leq T \times C_{p_{de}}, \tag{8}$$

where $PU_{p_{de}}$ is the predicted used capacity vector of PM p_{de} and PU_v is the predicted used capacity vector of VM v. The aggregated predicted resource utilization of VM and PM is bounded by a specified upper threshold T of total PM capacity. For example, if $T = 0.5$, the total predicted used utilization of PM p_{de} and VM v does not exceed more than 50% of total PM resource utilization. Based on above two constraints, the algorithm selects a destination PM that has enough available capacity in all dimension at the current time and the near future for allocating a VM. In summary, the algorithm proceeds by migrating all VMs from the least loaded PMs to the highest loaded PMs in the second step. A migration is feasible only if the destination PM has sufficient CPU and memory resources to allocate the candidate VM at the moment and near of future.

The pseudocode of Algorithm 1 creates a final migration plan by running two maintained steps. At the first step (lines 2-13), the algorithm looks through the set of overloaded PMs P_{over} and predicted overloaded PMs $\hat{P}_{over}$ (line 2). It aims to migrate some VMs from these sets to avoid more SLA violations. For this purpose, it first selects a VMs that needs a minimum time for migration. As we use 1Gbps network links between PMs, the migration time is only calculated based on the requested memory utilization by the VM. Therefore, the algorithm sorts all VMs of the source PM in ascending order of requested memory utilization and then starts to migrate some VMs (lines 3-4). If the source PM is still considered as a member of overloaded or predicted overloaded sets, the VM selection policy is applied again to select another VM to migrate from the PM (line 5). This is repeated until the PM is considered being not overloaded at the moment and future. Then, the algorithm finds the appropriate destination PM for reallocating the migrated VM v based on two proposed constraints (lines 6-7). Finally, the new VM placement is added to a migration plan $M1$ as a member (line 8). The migration plan is a set of 3-tuple $(p_{so}; v; p_{de})$, where the source PM p_{so}, the VM to be migrated v, and the destination PM p_{de}. If active PMs do not have sufficient resource to allocate the VM v, a dormant PM p can be switched to ON (lines 11-13).

At the second step (lines 14-30), the algorithm sorts active PMs in decreasing order based on their load levels (line 15). UP-VMC starts from the last PM of the list P_{active} and considers the least-loaded PM as a source PM p_{so} (lines 16-17). Then it aims to migrate all VMs from the PM p_{so} in order to release p_{so}. To select which VMs first migrate from p_s to other PMs, the algorithm sorts all VMs on PM p_s in decreasing order based on their load (line 18). To find the appropriate destination PM p_{de} for reallocating the migrated VMs, UP-VMC starts from the first PM (the most-loaded PM) of list P_{active} (line 19). If it is not possible, the second PM will be selected and so on. UP-VMC selects the PM p_{de} that has the required capacity for allocating the VM at the current time and near of future (lines 20). Finally, the new VM placement is added to a migration plan $M2$ as a member (line 23). The CPU used capacity of source and destination PMs are updated to reflect the impact of the migration (line 24). The *success* variable is defined to check whether all VMs form the source PM are migrated or not. Either all VMs from the source PM are migrated if one of them fails, none of them are migrated. Therefore, the algorithm removes all tuples in the migration plan and recovers the resource capacity of source and destination PMs if the value of *success* is false (lines 27-29). Otherwise, the source PM is switched to the sleep mode when all of its VMs migrate from it, that is, when the PMs no longer has any VMs (line 30). The output of the UP-VMC algorithm is the migration plan M that combines all migration tuples at two steps of UP-VMC (lines 31-32).

Algorithm 1: Utilization Prediction-aware VM Consolidation (UP-VMC)

```
1  Set $M1 \leftarrow \varnothing$;
2  for $p_{so} \in [P_{over}, \hat{P}_{over}]$ do
3      Sort VMs on PM $p_{so}$ and $V_m$ in ascending order based on $U_v^{mem}$;
4      for $v \in V_m$ do
5          if $(p_{so} \in P_{over}) || (p_{so} \in \hat{P}_{over})$ then
6              for $p_{de} \in P - [P_{over}, \hat{P}_{over}]$ do
7                  if $(U_{p_{de}} + U_v \leq T \times C_{p_{de}})$ and $(PU_{p_{de}} + PU_v \leq T \times C_{p_{de}})$ then
8                      $M1 \leftarrow M1 \cup \{(p_{so}, v, p_{de})\}$;
9                      Update $U_{p_{so}}$ and $U_{p_{de}}$;
10                     break
11             if $(p_{so} \in P_{over}) || (p_{so} \in \hat{P}_{over})$ then
12                 Switch on the dormant PM $p$;
13         else break;
14 Set $M2 \leftarrow \varnothing$;
15 Sort $P_{active}$ in descending of $Load_p$;
16 for $i = |P_{active}|$ to $1$ do
17     $p_{so} \leftarrow P_{active}[i]$;
18     $V_m \leftarrow$ sort VMs on PM $p_{so}, V_p$ in descending order of $Load_v$;
19     for $v \in V_m$ do
20         $success \leftarrow false$;
21         for $p_{de} \in P_{active} - p_{so}$ do
22             if $(U_{p_{de}} + U_v \leq T \times C_{p_{de}})$ and $(PU_{p_{de}} + PU_v \leq T \times C_{p_{de}})$ then
23                 $M2 \leftarrow M2 \cup \{(p_{so}, v, p_{de})\}$;
24                 Update $U_{p_{so}}$ and $U_{p_{de}}$;
25                 $success \leftarrow true$;
26                 break
27     if $success = false$ then
28         $M2 \leftarrow \varnothing$;
29         Recover $U_{p_{so}}$ and $U_{p_{de}}$;
30     else Switch $p_{so}$ to the sleep mode;
31 $M \leftarrow M1 \cup M2$;
32 return $M$
```

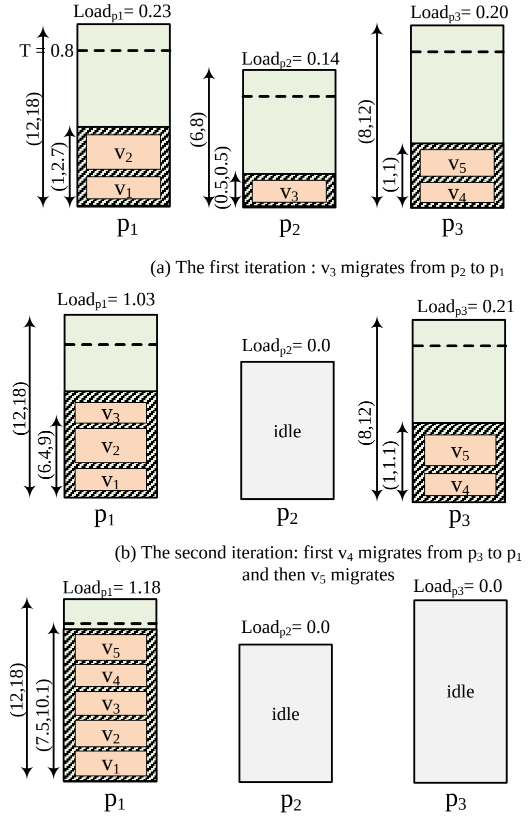
3.3 Example

Figure 2 shows an example of the UP-VMC algorithm where we have three heterogeneous PMs $P = \langle p_1, p_2, p_3 \rangle$ and five VMs $V = \langle v_1, v_2, v_3, v_4, v_5 \rangle$ are allocated on them. The total capacity vector of PMs: $C_{p_1} = \langle 12, 18 \rangle$, $C_{p_2} = \langle 6, 8 \rangle$ and $C_{p_3} = \langle 8, 12 \rangle$. Here $C_{p_1} = \langle 12, 18 \rangle$ means PM 1's CPU capacity and memory are 12GHz and 18GB, respectively. The total capacity vectors of VMs are : $C_{v_1} = \langle 1, 1 \rangle$, $C_{v_2} = \langle 1, 4 \rangle$, $C_{v_3} = \langle 2, 2 \rangle$, $C_{v_4} = \langle 2, 2 \rangle$ and $C_{v_5} = \langle 3, 4 \rangle$. For instance $C_{v_1} = \langle 1, 1 \rangle$ that means the VM 1's CPU and memory consumption are 1GHz and 1GB, respectively. We assume CPU threshold T = 0.8 in this example.

In the first iteration of the UP-VMC algorithm, the requested resource usages by VMs are: $U_{v_1} = \langle 0.5, 0.7 \rangle$, $U_{v_2} = \langle 0.5, 2 \rangle$, $U_{v_3} = \langle 0.5, 0.5 \rangle$, $U_{v_4} = \langle 0.5, 0.5 \rangle$ and $U_{v_5} = \langle 0.5, 0.5 \rangle$. Here $U_{v_1} = \langle 0.5, 0.7 \rangle$ means the v_1 requests 0.5 CPU capacity of 1 GHz and 0.7 memory of 1GB. Based on the initial VM placement (Figure 2a), the used capacity vector of PMs are estimated: $U_{p_1} = \langle 1, 2.7 \rangle$, $U_{p_2} = \langle 0.5, 0.5 \rangle$ and $U_{p_3} = \langle 1, 1 \rangle$. For example, $\langle 0.5, 0.7 \rangle$ be a tuple of the CPU and memory requests of the VM v_1, and $\langle 0.5, 2 \rangle$ is the requested capacity of the VM v_2. Then the utilization of PM p_1 accommodating the two VMs are estimated at $\langle 1, 2.7 \rangle$, i.e., the sum of the vectors. The UP-VMC algorithm first calculates the load of each PM by using Equation (3):

$$Load_{p_1} = \frac{U_{p_1}^{CPU}}{C_{p_1}^{CPU}} + \frac{U_{p_1}^{mem}}{C_{p_1}^{CPU}} = \frac{1}{12} + \frac{2.7}{18} \simeq 0.23.$$

Similarly the load of p_2 and p_3 are: $Load_{p_2} \simeq 0.14$ and $Load_{p_3} \simeq 0.20$. In the first step, the UP-VMC algorithm does not find any over-utilized and predicted over-utilized PMs. Because the current and predicted utilization of all PMs do not exceeds PMs'capacity. In the second step, algorithm



(a) The first iteration : v$_3$ migrates from p$_2$ to p$_1$

(b) The second iteration: first v$_4$ migrates from p$_3$ to p$_1$ and then v$_5$ migrates

(c) The third iteration: no migration

Fig. 2. An example of UP-VMC.

determines p_2 as a least loaded PM according to the current load of PMs, and aims to migrate v_3 in order to release p_2.

To find the destination PM for allocating v_3, UP-VMC starts from the most-loaded PM p_1 and, investigates both conditions in Equation (7) and (8) as

$$U_{p_1} + U_{v_3} \leq T \times C_{p_1} = \langle 1, 2.7 \rangle + \langle 0.5, 0.5 \rangle \leq \\ 0.8 \times \langle 12, 18 \rangle = \langle 1.5, 3.2 \rangle \leq \langle 9.6, 14.4 \rangle .$$

Since the PM p_1 has sufficient capacity in all dimensions for allocating the VM v_3, the first condition is satisfied. Moreover, the second condition is satisfied if the predicted capacity vector of the PM p_1 is $\langle 1.2, 2.5 \rangle$ and the predicted demand capacity vector of v_3 is $\langle 0.8, 1.3 \rangle$.

$$PU_{p_1} + PU_{v_3} \leq T \times C_{p_1} = \langle 1.2, 2.5 \rangle + \langle 0.8, 1.3 \rangle \leq \\ 0.8 \times \langle 12, 18 \rangle = \langle 2, 3.8 \rangle \leq \langle 9.6, 14.4 \rangle .$$

As both conditions are satisfied, the VM v_3 can migrate to the PM p_1 from the PM p_2. As the PM p_2 does not host any VMs, it switches to the sleep mode in order to reduce energy consumption.

In the second iteration of algorithm, algorithm does not run the first step because the active PMs p_1 and p_3 are not overloaded or predicted overloaded PMs. In the second step, assuming the current requested resource usages by VMs are: $U_{v_1} = \langle 1.4, 3 \rangle$, $U_{v_2} = \langle 2, 2 \rangle$, $U_{v_3} = \langle 3, 4 \rangle$, $U_{v_4} = \langle 0.5, 0.5 \rangle$ and $U_{v_5} = \langle 0.5, 0.6 \rangle$. According to the current VM placement (Figure 2(b)), the used capacity vectors of PMs are estimated: $U_{p_1} = \langle 6.4, 9 \rangle$, $U_{p_2} = \langle 0, 0 \rangle$ and $U_{p_3} = \langle 1, 1.1 \rangle$. The load level of active PMs are $Load_{p_1} \simeq 1.03$ and $Load_{p_3} \simeq 0.21$. As the load of the PM p_3 is less than the PM p_1, it can be switched to the sleep mode if the PM p_1 can accommodate the VM v_4 and v_5. UP-VMC first migrates a VM from the PM p_3 that has a maximum load level. The load level of VMs is measured by Equation (5) as:

$$Load_{v_4} = \frac{U^{CPU}_{v_4}}{C^{CPU}_{v_4}} + \frac{U^{mem}_{v_4}}{C^{mem}_{v_4}} = \frac{0.5}{2} + \frac{0.5}{2} \simeq 0.5,$$

$$Load_{v_5} = \frac{U^{CPU}_{v_5}}{C^{CPU}_{v_5}} + \frac{U^{mem}_{v_5}}{C^{mem}_{v_5}} = \frac{0.5}{3} + \frac{0.6}{4} \simeq 0.31.$$

Since the VM v_4 has higher load than the VM v_5, the UP-VMC algorithm tries to migrate the VM v_4 to the PM p_1. The PM p_1 requires to have a enough resource at the moment and future for allocation the VM. The destination PM is the PM p_1 if both following conditions are satisfied where $PU_{p_1} = \langle 1, 2.5 \rangle$ and $PU_{v_4} = \langle 0.5, 0.6 \rangle$:

$$U_{p_1} + U_{v_4} \leq T \times C_{p_1} = \langle 6.4, 9 \rangle + \langle 0.5, 0.5 \rangle \leq \\ 0.8 \times \langle 12, 18 \rangle = \langle 6.9, 9.5 \rangle \leq \langle 9.6, 14.4 \rangle ,$$

$$PU_{p_1} + PU_{v_4} \leq T \times C_{p_1} = \langle 1, 2.5 \rangle + \langle 0.5, 0.6 \rangle \leq \\ 0.8 \times \langle 12, 18 \rangle = \langle 1.5, 3.1 \rangle \leq \langle 9.6, 14.4 \rangle .$$

Therefore, the VM v_4 can migrate to p_1 because it has enough capacity to reallocate the VM at the moment and near of future. Then the algorithm also check whether p_1 has sufficient capacity for allocating v_5 by following conditions, assume $PU_{v_5} = \langle 0.7, 0.5 \rangle$:

$$U_{p_1} + U_{v_5} \leq T \times C_{p_1} = \langle 6.4, 9 \rangle + \langle 0.5, 0.6 \rangle \leq \\ 0.8 \times \langle 12, 18 \rangle = \langle 6.9, 9.6 \rangle \leq \langle 9.6, 14.4 \rangle ,$$

$$PU_{p_1} + PU_{v_5} \leq T \times C_{p_1} = \langle 1, 2.5 \rangle + \langle 0.7, 0.5 \rangle \leq \\ 0.8 \times \langle 12, 18 \rangle = \langle 1.7, 3 \rangle \leq \langle 9.6, 14.4 \rangle .$$

So the VM v_5 can migrate to p_1 and after migration the PM p_3 can be switched to the sleep mode. Finally the number of PMs is reduced from 3 to 1. However, two released PMs can be switched to on if the amount of resource requests increases.

4 Experimental Setup

In this section, first we describe the simulation setup for our proposed approach evaluation. Then, we explain two type of workloads (PlanetLab and Google) and evaluation metrics. Finally, we describe the benchmark algorithms that are implemented to compare with the proposed approach.

4.1 Simulation Setup

To evaluate the efficiency of our proposed VM consolidation approach, we use the CloudSim toolkit [23]. We simulated a data center comprising M heterogeneous PMs. The value of M depends on the type of workload which is specified in Table 2. In each workload, half of PMs are HP ProLiant ML110 G4 servers 1,860 MIPS each core, and the other half consists of HP ProLiant ML110 G5 servers with 2,660 MIPS each core. Each PM is modeled to have 2 cores, 4GB memory and 1 GB/s network bandwidth. The CPU MIPS rating and the memory amount characteristics of four VM instances used in CloudSim corresponded to Amazon EC2 [1], i.e., 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 2. Characteristics of the considered workload traces.

Workload	Date	VMs	Res.	Mean (%)	St. dev (%)	Median
PlanetLab	May - Apr. 2011	265	CPU	25.44	14.16	22
			Mem	10.48	11.06	7
GCD	10 days May 2011	1,600	CPU	10.87	10.85	7
			Mem	22.87	16.05	20

TABLE 3. The energy consumption at different load levels in Watts.

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

4.2 Workload Data

To make the simulation based evaluation applicable, we evaluated the UP-VMC approach on real-world publicly available workloads:

- PlanetLab data [24]: is provided as a part of the CoMon project, a monitoring infrastructure for PlanetLab. In this project, the CPU and memory usage data is reported every five minutes from more than a thousand VMs and is stored in ten 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. We consider ten days from the PlanetLab VMs workload traces collected during March and April 2011. As the utilization value of some VMs in the data set is so low, we filter the original data and only consider the range of CPU and memory utilization between 5% to 90%. So we can evaluate the proposed VM consolidation by considering both CPU and memory intensive tasks.
- Google Cluster Data (GCD) [25]: provides real trace data of a Google cluster over about one-month period in May 2011. This trace involves over 650 thousand jobs across over 12,000 heterogeneous PMs. Each jobs with one or more tasks, contains the normalized value of the average number of used cores and the utilized memory. The usage of each type of resources is collected at five minutes intervals. For our experiments, we extracted the task duration based on the time when the task was scheduled last and the time when the task finished. Furthermore, we also extracted the task utilization values of CPU and memory over the first ten days. We use the jobID as the unique identifier for a job, and for each of these jobs we extracted a set of actual usage for each resource for all of its tasks. The attributes that we considered for CPU and memory are: the CPU rate, which indicates the average CPU utilization for a sample period of 5 minutes, and the canonical memory usage, which represents the average memory consumption for the same sampling period. Moreover, tasks with a CPU or RAM consumption higher than 90% and lower than 5% where also removed from the experiments.

The characteristics of the VMs and their resource utilization in the PlanetLab and GCD traces are presented in Table 2.

4.3 Performance 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 [12] that can be used to evaluate the SLA delivered by any VM deployed in an IaaS. SLAV is measured 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, both performance degradation due to host overloading and due to VM migrations are proposed as a combined metric (SLAV)

$$SLAV = SLAVO \times SLAVM. \tag{9}$$

  In this paper, SLAVO indicates the percentage of time, during which active PMs have experienced the CPU or memory utilization of 100% as

$$SLAVO = \frac{1}{M}\sum_{i=1}^{M}\frac{T_{s_i}}{T_{a_i}}, \tag{10}$$

  where M is the number of PMs; T_{s_i} is the total time that the PM i has experienced the CPU or memory 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

$$SLAVM = \frac{1}{N}\sum_{j=1}^{N}\frac{C_{d_j}}{C_{r_j}}, \tag{11}$$

  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 [12], [26] 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2617374, IEEE Transactions on Cloud Computing

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 3 illustrates the amount of energy consumption of two types of HP G4 and G5 servers at different load levels. The table shows the energy consumption is reduced efficiently when under-utilized PMs switch to the sleep mode.

- **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 [11]. In addition, VM migration consumes negligible energy that is considered for measuring SLA violations as 10% of CPU utilization during all VM migrations in the servers.

4.4 Baseline Algorithms

We implemented and evaluated the Utilization Prediction-aware VM Consolidation (UP-VMC) approach on a simulated data center. We discuss the added value of employing an utilization prediction model by comparing UP-VMC with the following algorithms:

- Modified Best Fit Decreasing (MBFD): in order to applicable comparison with UP-VMC, we modified the traditional BFD algorithm (MBFD) by proposing a threshold to limit the resource utilization of a PM for allocating VMs. Thus MBFD assigns VM v to a most-loaded PM p_{de} if the total used capacity of the VM and PM does not exceed the threshold of the total resource utilization of the PM:
$$U_{p_{de}} + U_v \leq T \times C_{p_{de}}. \tag{12}$$
In comparison with the UP-VMC algorithm, MBFD dose not employ any prediction models so that it only migrates VMs from the over-loaded PMs to avoid SLA violations. Furthermore, MBFD can reduce the energy consumption by releasing least-loaded PMs.
- Modified First Fit Decreasing (MFFD): is a modified version of FFD algorithm to compare with the UP-VMC algorithm. The MFFD allocates a VM to the first PM that the total used capacity vector of the VM and PM is less than the threshold of the total capacity vector of the PM.
$$U_{p_{de}} + U_v \leq T \times C_{p_{de}}. \tag{13}$$
In addition, MFFD minimizes SLA violations by reallocating VMs from the over-loaded PMs. It also performs the second step of UP-VMC to minimize the number of active PMs based on the current resource requirements.
- PM Utilization Prediction-aware VM Consolidation (PUP-VMC): runs an algorithm similar to the UP-VMC algorithm but it only considers the future resource utilization of a destination PM to allocate a VM. Therefore, PUP-VMC allocates the VM v to the PM p_{de} if the aggregated the predicted capacity vector of PM p_{de} and the demand capacity vector VM v does not exceed of the threshold of the total capacity vector of the PM:
$$PU_{p_{de}} + U_v \leq T \times C_{p_{de}}. \tag{14}$$
- VM Utilization Prediction-aware VM Consolidation (VUP-VMC): runs an algorithm similar to the UP-VMC algorithm but it only considers the future demand capacity of a VM. The VM v can be assigned to the destination PM p_{de} if the aggregated used capacity vector of the PM and the predicted demand capacity vector by the VM does not exceed of the threshold of the total capacity vector of the PM:
$$U_{p_{de}} + PU_v \leq T \times C_{p_{de}} \tag{15}$$
- Sercon [11]: is a well-known greedy bin-packing algorithm. It migrates VMs from the least loaded PM to the most loaded PM to release the least loaded PM. In order to eliminate unnecessary migration, Sercon follows all-or-nothing property, that is, either all VMs from a PM are migrated or if one of them fails, none of them are migrated.
- ACS-based VM Consolidation (ACS-VMC) [5]: is a meta-heuristic bin-packing algorithms. It generates a migration plan using ant colony system algorithm to reduce the energy consumption and SLA violations.

5 Experimental Results

In this section, we first present the impact of three different VM selection polices on the UP-VMC. Then, the prediction performance is evaluated. Finally, we discuss about the experimental results in comparison with the benchmark algorithms.

5.1 VM Selection Algorithms

We investigate the impact of three well-known VM selection polices on the UP-VMC performance. These policies include:

- Minimum Migration Time (MMT): a VM with the minimum migration time is selected for migration. The length of a VM migration takes as long as it needs to migrate the memory assigned to the VM over the network bandwidth link between source and destination PMs. In our simulations, we used 1Gbps network links.
- Maximum Load (MaxL): a VM with the maximum load level ($Load_v$) is selected for migration.
- Minimum Load (MinL): a VM with the minimum load level ($Load_v$) is selected for migration.

The effectiveness of the VM selection polices is evaluated based on the three metrics: the number of SLA violations, the energy consumption and the number of migrations. As depicted in Figure 3 and Figure 4, UP-VMC reduced the energy consumption, SLA violations and the number of migrations when it uses MMT policy. In addition, the results show UP-VMC can perform better in terms of energy consumption, SLA violations and number of migrations

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2617374, IEEE Transactions on Cloud Computing

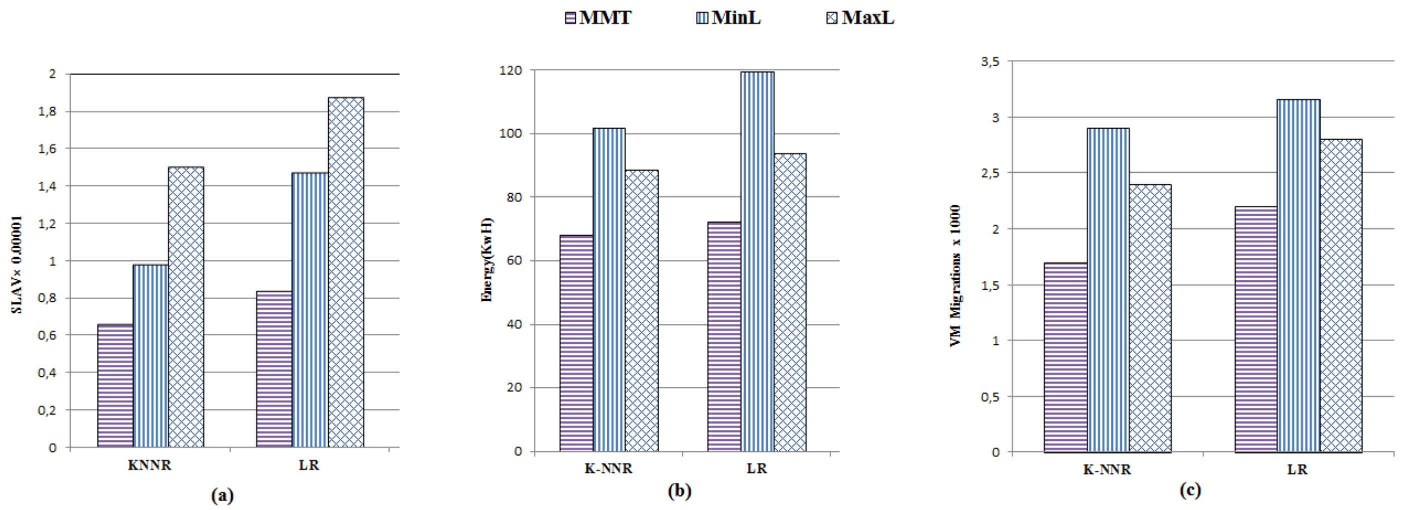


Fig. 3. The SLAV metric (a), energy consumption (b), and number of VM migrations (c) by UP-VMC based on LR and K-NNR with three VM selection polices for the PlanetLab workload trace.

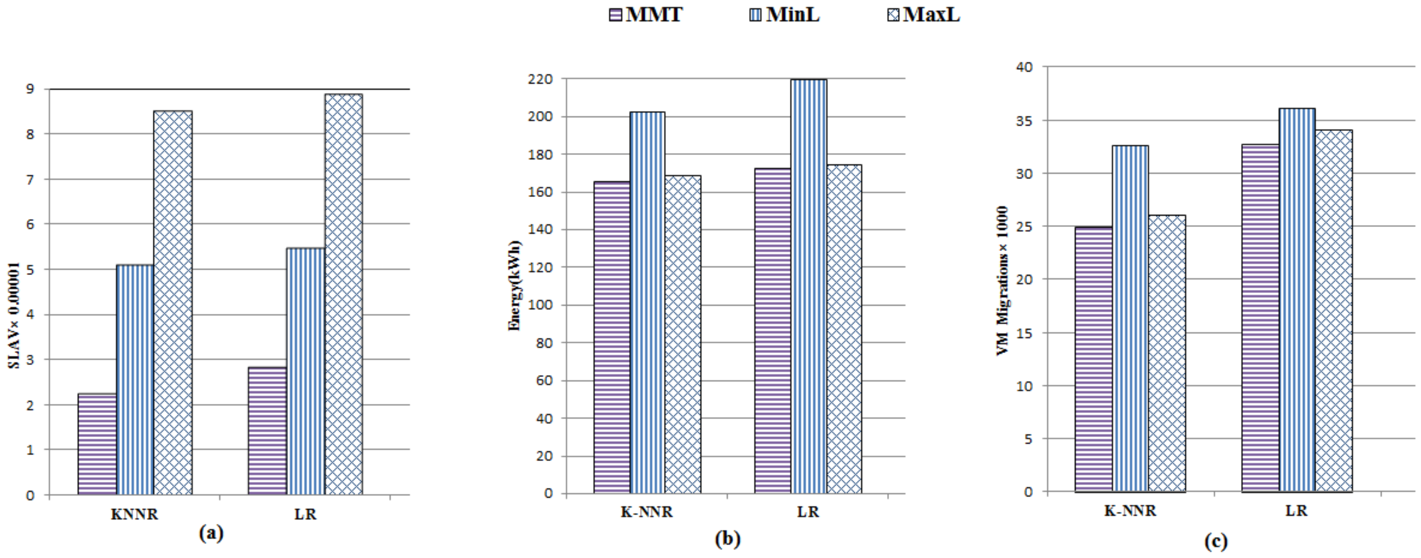
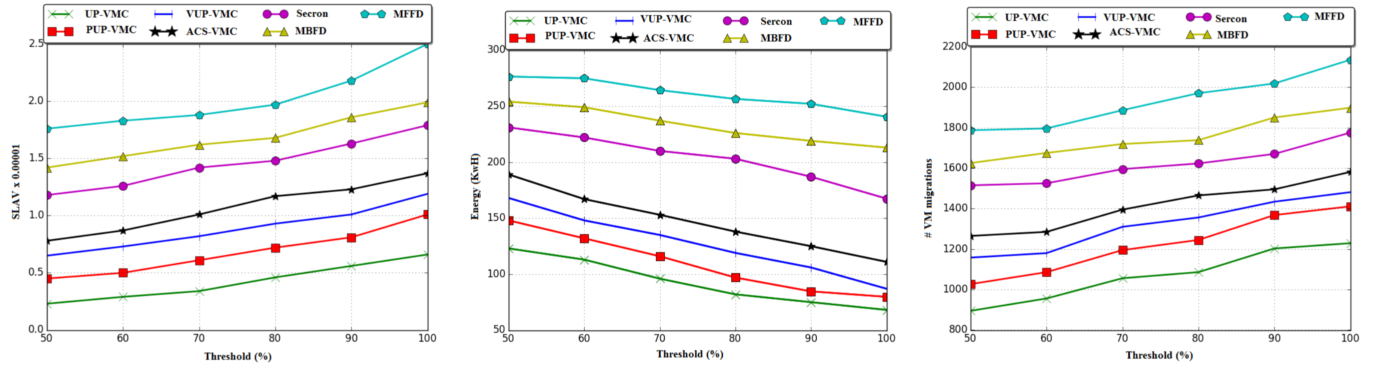
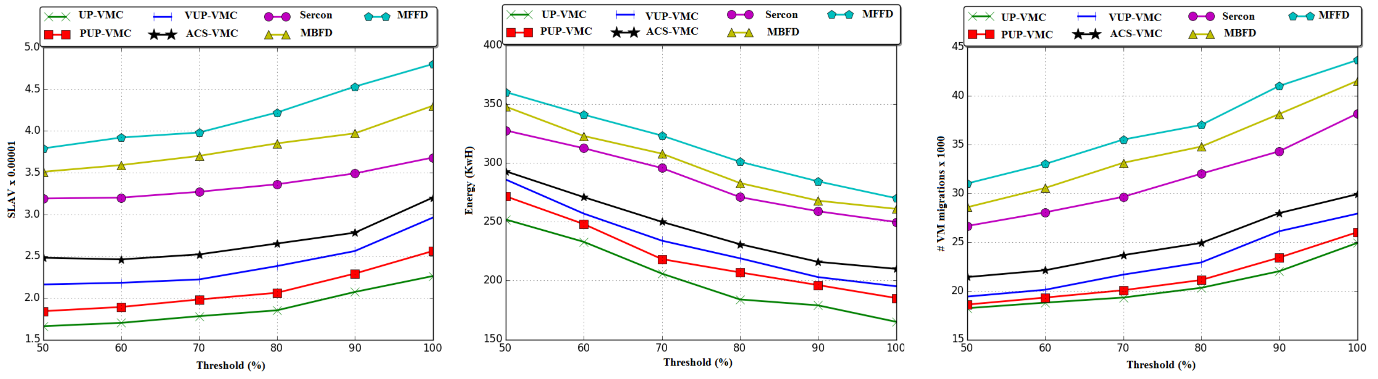


Fig. 4. The SLAV metric (a), energy consumption (b) and number of VM migrations (c) by UP-VMC based on LR and K-NNR with three VM selection polices for the GCD workload trace.



(a) (b) (c)

Fig. 5. The SLAV metric (a), energy consumption (b), and number of VM migrations (c) by UP-VMC and benchmark methods for the PlanetLab workload trace.



(a)

(b)

(c)

Fig. 6. The SLAV metric (a), energy consumption (b), and number of VM migrations (c) by UP-VMC and benchmark methods for the GCD workload trace.

when it uses K-NNR as a prediction model. This is because K-NNR able to accurately measure the resource utilization of PMs and VMs than LR (see next sub-section).

5.2 Performance Evaluation of Prediction models

We applied the Leave-One-Out cross-validation technique [27] for evaluating the accuracy of the prediction model. We then assess the prediction model accuracy by using Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE). The first measures the average of the absolute errors, where the errors are calculated as the difference between the predicted and the actual value. The latter measures the standard deviation of the absolute errors. To evaluate the quality of our prediction technique, we used five different data sets. Each data set collects the resource utilization of a PM by running GCD workload traces. Thus a one-step prediction is equivalent to predict the PM 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.

The results show RMSE of K-NN is 0.21 where it is 0.38 for LR. Moreover, MAPE of K-NN and Linear Regression (LR) prediction models are 0.24 and 0.37, respectively. So MAPE of the K-NN is slightly lower than the linear regression which means that the K-NN can predict the actual utilization more accurately by yielding less residuals. We also conducted more experiments to examine the impact of prediction step on the prediction performance. Experiments results show the prediction error is grown when we increased the prediction step. Therefore, we predict the future load of PM in the next time.

5.3 Comparison of UP-VMC with Baseline Algorithms

We conducted several experiments to evaluate the performance of our approach. In the first experiments, we use PlanetLab workload. As 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, finding a suitable threshold value is essential. Therefore we obtained results on various threshold values between 50% to 100% to show the impact of the value of threshold on the performance metrics. Figure 5a presents the SLA violation levels caused by the UP-VMC, PUP-VMC, VUP-VMC, ACS-VMC, Sercon, MBFD and MFFD methods for different values of threshold. The UP-VMC 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-VMC allocates a VM to a PM that does not become overloaded in the near future based on the proposed prediction model. In addition, UP-VMC tries to migrate VMs from the overloaded and predicted overloaded PMs in order to avoid the SLA violations. Figure 5b shows our proposed VM consolidation approach can reduce energy consumption by up to 71.6% in comparison with MFFD. This is because the proposed method minimize the number of active PMs by packing VMs into the most-loaded PMs. Figure 5c depicts the total number of VM migrations during the VM consolidation in the PlanetLab workload. The UP-VMC 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 GCD with the large number of VMs. During the simulation, each VM is randomly assigned a workload trace from one of the VMs from the data set. Figure 6a shows that the UP-VMC leads to significantly less SLA violations than the other four benchmark algorithms. This is due to the fact that the UP-VMC prevents SLA violations by using a prediction of PM and VM utilization and ensures that the destination PM does not become overloaded when a VM migrates on it. Figure 6b illustrates the UP-VMC brought higher energy savings in comparison to the other approaches in the real workload. This is because the UP-VMC tries to release least-loaded PMs by packing VMs into a few number of PMs. As observed from the results, the UP-VMC has the minimum number of migrations compared with the other benchmark methods (Figure 6c). This is due to the fact it allocates a VM to a PM according to the current and future resource utilization.

Moreover, the effect of different threshold values on three evaluation metrics is shown in Figure 5 and Figure 6. These figures show that the increase of threshold leads to the decrease the energy consumption, indicating that although a smaller value of threshold achieves better performance on energy reduction. Moreover, the number of migrations and SLA violations are increased if the threshold is set in a high value. Therefore, it seems that a good solution is achievable by properly selecting the value of threshold.

6 Conclusion and Future Work

In this paper, we presented a dynamic Virtual Machine (VM) consolidation approach called Utilization Prediction-aware VM Consolidation (UP-VMC). UP-VMC formulates a VM consolidation as a multi-objective vector bin packing problem. It considers both the current and future utilization of resources in order to consolidate VMs into the minimum number of active Physical Machines (PMs). The future resource utilization is predicted by using a regression-based prediction model. To further enhance the quality of service and minimize the number of migrations, we also proposed a VM allocation algorithm based on prediction models. The algorithm selects a PM that has enough resources at the moment and in the short term of future for reallocating the VM. The obtained results of real Google and PlanetLab workload traces show that UP-VMC significantly outperforms benchmark algorithms in terms of energy consumption, performance requirements and number of migrations.

As a future work, we have identified four improvement directions for the UP-VMC. First, we plan to improve the scalability of the proposed architecture by moving from two to three-tier hierarchical architecture. The second improvement, UP-VMC will take into account network resource utilization and traffic to optimize VM placement. The third improvement aims to implement UP-VMC algorithm as an extension of the VM manager within the OpenStack Cloud platform[2] to evaluate the algorithm in a real cloud environment. The four improvement involves evaluating the performance of the proposed VM consolidation algorithm over different thresholds for different resources types.

References

[1] P. Barham, B. Dragovic, K. Fraser, S. H, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Ninteenth acm Symposium on Operating Systems Principles(SOSP)*, 2003, pp. 164–177.

[2] VMwareInc., "How vmware virtualization right-sizes it infrastructure to reduce power consumption," 2009.

[3] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *2nd conference on Symposium on Networked Systems Design and Implementation*, vol. 2, 2005, pp. 273–286.

[4] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in clouds: A performance evaluation," in *1st International Conference on Cloud Computing (CloudCom)*. Springer-Verlag, 2009, pp. 254–265.

[5] 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.

[6] E. Feller, C. Morin, and A. Esnault, "A case for fully decentralized dynamic vm consolidation in clouds," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, 2012, pp. 26–33.

[7] F. Farahnakian, P. Liljeberg, and J. Plosila, "Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning," in *22nd Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, 2014, pp. 500–507.

[8] F. Farahnakian, A. Ashraf, P. Liljeberg, T. Pahikkala, J. Plosila, I. Porres, and H. Tenhunen, "Energy-aware dynamic vm consolidation in cloud data centers using ant colony system," in *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, 2014, pp. 104–111.

[9] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Computer Networks*, vol. 53, pp. 2923–2938, 2009.

[10] 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.

[11] A. Murtazaev and S. Oh, "Sercon: Server consolidation algorithm using live migration of virtual machines for green computing," vol. 28, 2011, pp. 212–231.

[12] A. Beloglazov and Buy, "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 (CCPE)*, vol. 24, pp. 1397–1420, 2012.

[13] 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.

[14] 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.

[15] E. Caron, F. Desprez, and A. Muresan, "Forecasting for grid and cloud computing on-demand resources based on pattern matching," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, Nov 2010, pp. 456–463.

[16] C. Chekuri and S. Khanna, "On multi-dimensional packing problems," in *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '99. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1999, pp. 185–194. [Online]. Available: http://dl.acm.org/citation.cfm?id=314500.314555

[17] 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.

[18] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, and H. Tenhunen, "Utilization prediction aware vm consolidation approach for green cloud computing," in *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, 2015, pp. 381–388.

[19] 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.

[20] M. Stillwell, D. Schanzenbach, F. Vivien, and H. Casanova, "Resource allocation algorithms for virtualized service hosting platforms," *J. Parallel Distrib. Comput.*, vol. 70, no. 9, pp. 962–974, Sep. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.jpdc.2010.05.006

[21] J. Han, *Data Mining: Concepts and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[22] F. J. Ferrer-Troyano, J. S. Aguilar-Ruiz, and J. C. Riquelme, "Empirical evaluation of the difficulty of finding a good value of k for the nearest neighbor," in *2003 International Conference on Computational Science*. Springer-Verlag, 2003, pp. 766–773.

[23] R. 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 (SPE)*, vol. 41, pp. 23 – 50, 2011.

[24] K. Park and V. S. Pai, "Comon: A mostly-scalable monitoring system for planetlab," vol. 40, 2006, pp. 65–74.

[25] "Traces of google workloads," 2015, http://code.google.com/p/googleclusterdata/.

[26] 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.

[2] $http://openstack.org/$

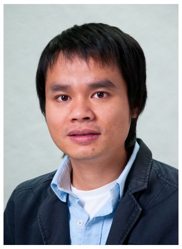
[27] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," *Statistics Surveys*, vol. 4, pp. 40–79, 2010.

Fahimeh Farahnakian received M.S degree in Computer Engineering from the University of Science and Technology, Tehran, Iran. She is currently pursuing her Ph.D. in Embedded Computer and Electronic Systems Laboratory, University of Turku, Finland. Her research interests include cloud computing, energy efficient data center, network on- chips, machine learning. She is a member of IEEE and is a frequent reviewer for research journals, such as Journal of Circuits, Systems and Computers (JCSC), International Journal of High Performance Systems Architecture (IJHPSA), Journal of Low Power Electronics (JOLPE) and Journal of Supercomputing.

Tapio Pahikkala currently acts as a professor of intelligence systems in University of Turku, Finland. His research focuses on machine learning, pattern recognition, algorithmic, and computational intelligence. He has authored more than 80 peer-reviewed scientific publications and served in program committees of numerous scientific conferences. He is a member of both ACM and IEEE, and he served as member in several committees of IEEE societies.

Pasi Liljeberg received his M.Sc. and Ph.D. degrees in electronics and information technology from the University of Turku, Turku, Finland, in 1999 and 2005, respectively. He is an Associate Professor in Embedded Electronics laboratory and an Adjunct Professor in embedded computing architectures at the University of Turku, Embedded Computer Systems laboratory. Liljeberg is the author of over 150 peer- reviewed publications, has supervised 9 PhD theses. His current research interests include parallel and distributed systems, Internet-of-Things, embedded computing architecture, fault tolerant and energy aware system design.

Juha Plosila is an Associate Professor in Embedded Computing and an Adjunct Professor in Digital Systems Design at the University of Turku, Department of Information Technology, Finland. Dr. Plosila is the leader of the Embedded Computer Systems laboratory and a co-leader of the Resilient IT Infrastructures research programmer at Turku Centre for Computer Science. Dr.Plosila is an Associate Editor of International Journal of Embedded and Real-Time Communication Systems (IGI Global).



Nguyen Trung Hieu received his M.S. degree from the Department of Computer Engineering at the Kyung Hee University, South Korea, in 2012. He is currently pursuing his Ph.D. at the Department of Computer Science, Aalto University, Finland. His current research interests include distributed processing, cloud computing and energy-efficient data centers.

Hannu Tenhunen received the Diplomas from the Helsinki University of Technology, Finland, 1982, and the PhD degree from Cornell University , Ithaca, New York, 1986. Since 1992, ha has been a professor at the Royal Institute of Technology (KTH), Sweden, where he also served as dean. He is currently the director of Turku Center for Computer Science , Finland, and at the University of Turku. He has more than 600 reviewed publications and 16 patents internationality. He is a member of the IEEE.