Using Ant Colony System to Consolidate VMs for Green Cloud Computing

Fahimeh Farahnakian, *Member, IEEE*, Adnan Ashraf, Tapio Pahikkala, *Member, IEEE*, Pasi Liljeberg, Juha Plosila, Ivan Porres, and Hannu Tenhunen, *Member, IEEE*

Abstract—High energy consumption of cloud data centers is a matter of great concern. Dynamic consolidation of Virtual Machines (VMs) presents a significant opportunity to save energy in data centers. A VM consolidation approach uses live migration of VMs so that some of the under-loaded Physical Machines (PMs) can be switched-off or put into a low-power mode. On the other hand, achieving the desired level of Quality of Service (QoS) between cloud providers and their users is critical. Therefore, the main challenge is to reduce energy consumption of data centers while satisfying QoS requirements. In this paper, we present a distributed system architecture to perform dynamic VM consolidation to reduce energy consumption of cloud data centers while maintaining the desired QoS. Since the VM consolidation problem is strictly NP-hard, we use an online optimization metaheuristic algorithm called Ant Colony System (ACS). The proposed ACS-based VM Consolidation (ACS-VMC) approach finds a near-optimal solution based on a specified objective function. Experimental results on real workload traces show that ACS-VMC reduces energy consumption while maintaining the required performance levels in a cloud data center. It outperforms existing VM consolidation approaches in terms of energy consumption, number of VM migrations, and QoS requirements concerning performance.

Index Terms—Dynamic VM consolidation, ant colony system, cloud computing, green computing, energy-efficiency, SLA

1 Introduction

Cloud computing is a relatively new computing paradigm. It leverages several existing concepts and technologies, such as data centers and hardware virtualization, and gives them a new perspective. Cloud computing provides three service models and four deployment models [1]. The three service models are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Similarly, the four deployment models are private cloud, community cloud, public cloud, and hybrid cloud [2], [3]. With its pay-per-use business model for the customers, cloud computing shifts the capital investment risk for under or over provisioning to the cloud providers [4], [5]. Therefore, several public IaaS, PaaS, and SaaS cloud providers, such as Amazon, Google, and Microsoft, operate large-scale cloud data centers around the world. Moreover, due to the ever-increasing cloud infrastructure demand, there has been a significant increase in the size and energy consumption (EC) of the cloud data centers [6]. High energy consumption not only translates to a high operating cost, but also leads to higher carbon emissions. Therefore, 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. On the other hand, achieving the desired level of Quality of Service (QoS) between cloud providers and their users is critical for satisfying customers' expectations concerning performance. The QoS requirements are formalized via Service Level Agreements (SLAs) that describe the required performance levels, such as minimal throughput and maximal response time or latency of the system. Therefore, the main challenge is to reduce energy consumption of data centers while satisfying QoS requirements.

Over the past few years, there have been several attempts to reduce energy consumption of data centers. Currently, the two widely-used techniques are dynamic server provisioning and Virtual Machine (VM) consolidation. Dynamic server provisioning approaches [7] save energy by using a reduced amount of resources needed to satisfy the workload requirements. Therefore, unnecessary servers are switched-off or put into a low-power mode when the workload demand decreases. Similarly, when the demand increases, additional servers are switched-on or put back into a high-power mode. Dynamic VM consolidation is another effective way to improve the utilization of resources and their energy-efficacy [6], [8]. It leverages the hardware virtualization technology [9], which shares a Physical Machine (PM) among multiple performance-isolated platforms called VMs, where each VM runs one or more application tasks. The sharing of the PM resources among multiple VMs is handled by the Virtual Machine Monitor (VMM). Therefore, virtualization takes dynamic server provisioning one step further and allows different applications to be allocated on the same PM to improve the resource utilizations. Moreover, it allows live VM migration and consolidation to pack VMs on a reduced number of PMs, reducing the energy consumption [10]. However, in order to maximize resource utility, it is essential to manage PM resources in an adequate manner. Therefore,

- *F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, and H. Tenhunen are with the Department of Information Technology, University of Turku, Turku 20520, Finland. E-mail: {fahfar, aatapa, pakrli, juplos, hanten}@utu.fi.*
- *A. Ashraf and I. Porres are with Department of Information Technologies, Åbo Akademi University, Turku 20520, Finland. E-mail: {aashraf, iporres}@abo.fi.*

Manuscript received 1 Nov. 2014; revised 3 Mar. 2015; accepted 2 Dec. 2014. Date of publication 28 Dec. 2014; date of current version 10 Apr. 2015. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TSC.2014.2382555

one of the most important optimization problems concerning VM consolidation is energy-efficient placement of VMs on PMs. Furthermore, to be able to cope with the workload variability of different types of applications, the VM consolidation should be performed in an online manner.

In this paper, we address the VM consolidation problem with the objective to reduce energy consumption of data centers while satisfying QoS requirements. We present a distributed system architecture to perform dynamic VM consolidation to improve resource utilizations of PMs and to reduce their energy consumption. We also propose a dynamic VM consolidation approach that uses a highly adaptive online optimization metaheuristic algorithm called Ant Colony System (ACS) [11], [12] 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. The performance of the proposed ACS-VMC approach is evaluated by using CloudSim [13] simulations on real workload traces, which were obtained from more than a thousand VMs running on servers located at more than 500 places around the world. The simulation results show that ACS-VMC maintains the desired QoS while reducing energy consumption in a cloud data center. It outperforms existing VM consolidation approaches in terms of energy consumption, number of VM migrations, and number of SLA violations (SLAV).

The remainder of this paper is organized as follows. Section 2 discusses some of the most important related works and briefly reviews the Ant Colony Optimization (ACO) metaheuristic. Section 3 and Section 4 present the system architecture and the proposed dynamic VM consolidation approach, respectively. Section 5 describes the experimental design and setup. Finally, we present the experimental results in Section 6 and our conclusions in Section 7.

2 Background and Related Work

The existing VM consolidation approaches, such as [14], [15], [16], [17] are used in data centers to reduce under-utilization of PMs and to optimize their energy-efficiency. The main idea in these approaches is to use live VM migration [10] to periodically consolidate VMs so that some of the underloaded PMs can be released for termination. Determining when it is best to reallocate VMs from an overloaded PM is an important aspect of dynamic VM consolidation that directly influences the resource utilization and QoS. In [18], two static thresholds were used to indicate the time of VM reallocation. This approach keeps the total Central Processing Unit (CPU) utilization of a PM between these thresholds. However, setting static thresholds is not efficient for an environment with dynamic workloads. Therefore, Beloglazov and Buyya [19] presented adaptive thresholds that can be derived based on the statistical analysis of the historical data.

Another important aspect of dynamic VM consolidation concerns load prediction on a PM. Using a prediction of the future load enables proactive consolidation of VMs on the overloaded and under-loaded PMs. Therefore, in our previous works [20], [21], we proposed two regression methods to predict CPU utilization of a PM. These methods use the linear regression and the K-nearest neighbor (KNN) regression algorithms, respectively, to approximate a function based on the data collected during the lifetimes of the VMs. Therefore, we used the function to predict an overloaded or an under-loaded PM for reducing the SLA violations and energy consumption.

The VM consolidation problem is known to be NP-hard [15], [22]. Therefore, it is expensive to find an optimal solution with a large number of PMs and VMs. In some of the existing approaches, VM consolidation has been formulated as an optimization problem with the objective to find a near optimal solution by using a greedy approach [22], [23], [24]. Since an optimization problem is associated with constraints, such as data center capacity and SLA, these works use a heuristic to consolidate workload in a multi-dimensional bin packing problem. The PMs are assumed to be the bins and the VMs are considered as the objects. The objective of bin packing is to minimize the number of bins while packing all the objects. Wood et al. [22] used a greedy algorithm to determine a sequence of moves to migrate overloaded VMs to under-loaded PMs. Ajiro and Tanaka [23] used a load balancing algorithm called least-loaded (LL) to balance the load among PMs. Wang et al. [24] formulated the VM consolidation problem as a stochastic bin packing problem and used an online packing algorithm to consolidate VMs with dynamic bandwidth demands.

In this paper, we formulate energy-efficient VM consolidation as a multi-objective combinatorial optimization problem and apply a highly adaptive online optimization [25] metaheuristic called Ant Colony Optimization [11] to find a near-optimal solution. ACO is a multi-agent approach to difficult combinatorial optimization problems, such as traveling salesman problem (TSP) and network routing [12]. It is inspired by the foraging behavior of real ant colonies. While moving from their nest to a food source and back, ants deposit a chemical substance on their path called pheromone. Other ants can smell pheromone and they tend to prefer paths with a higher pheromone concentration. Thus, ants behave as agents who use a simple form of indirect communication called *stigmergy* to find better paths between their nest and the food source. It has been shown experimentally that this simple pheromone trail following behavior of ants can give rise to the emergence of the shortest paths [12]. It is important to note here that although each ant is capable of finding a complete solution, high quality solutions emerge only from the global cooperation among the members of the colony who concurrently build different solutions. Moreover, to find a high quality solution, it is imperative to avoid *stagnation*, which is a premature convergence to a suboptimal solution or a situation where all ants end up finding the same solution without sufficient exploration of the search space [12]. In ACO metaheuristic, stagnation is avoided mainly by using pheromone evaporation and stochastic state transitions.

There are a number of ant algorithms, such as Ant System (AS), Max-Min AS (MMAS), and Ant Colony System [11], [12]. ACS was introduced to improve the performance of AS and it is currently one of the best performing ant algorithms. The existing ACO-based resource allocation and server consolidation approaches include [15], [26], [27], [28].

Yin and Wang [26] applied ACO to the nonlinear resource allocation problem, which seeks to find an optimal allocation of a limited amount of resources to a number of tasks to optimize their nonlinear objective function. Feller et al. [15] applied MMAS to the VM consolidation problem in the context of cloud computing. A recent paper by Ferdaus et al. [27] integrated ACS with a vector algebra-based server resource utilization capturing technique [29]. Another recent work by Ashraf and Porres [28] used ACS to consolidate multiple web applications in a cloud-based shared hosting environment.

In this paper, we apply ACS to the VM consolidation problem. Our main contributions are as follows:

- We formulate energy-efficient VM consolidation as a multi-objective combinatorial optimization problem to optimize three conflicting objectives simultaneously. The objectives include reducing energy consumption, minimizing the number of VM migrations, and avoiding SLA violations.
- We present a distributed multi-agent system architecture for dynamic VM consolidation. In our approach, a local agent (LA) detects PM status: normal, overloaded, predicted overloaded, or under-loaded. We use the LiRCUP method [20] to predict an overloaded PM for avoiding SLA violations, as described in Section 3. Moreover, a global agent (GA) dynamically consolidates VMs into a reduced number of PMs by using our proposed ACS-based VM Consolidation algorithm, which is presented in Section 4.
- We take into account the multi-dimensional resource utilizations of a PM. Therefore, VM consolidation in ACS-VMC is based on three resource dimensions: CPU, memory, and network Input/Output (I/O).
- The performance of the proposed ACS-VMC approach is evaluated by CloudSim simulation on real workload traces. We compared our proposed approach with the existing dynamic VM consolidation approaches in the CloudSim toolkit and with the ACS-based VM consolidation approach in [27]. The results show that ACS-VMC outperforms existing VM consolidation approaches in terms of energy consumption, number of VM migrations, and number of SLA violations.

3 System Architecture

A cloud data center consists of m heterogeneous PMs that have different resource capacities. Each PM contains a CPU, which is often a multi-core. The CPU performance can be defined in terms of Millions of Instructions Per Second (MIPS). In addition, a PM is also characterized by the amount of memory, network I/O, and storage capacity. At any given time, a cloud data center usually serves many simultaneous users. Users submit their requests for provisioning of n VMs, which are allocated to the PMs. The length of each request is specified in millions of instructions (MI). In our proposed approach, 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 [16]. BFD first sorts all VMs by their utilization weights in the decreasing order. Then, it starts with the VMs that require the largest amount of resources.

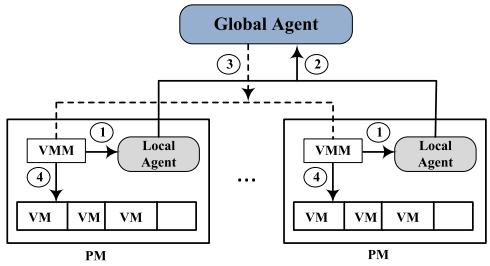


Fig. 1. The system architecture.

The BFD algorithm allocates VMs in such a way that the unused capacity in the destination PMs is minimized. Thus, it selects a PM for which the amount of available resources is closest to the requested amount of resources by the VM. Therefore, BFD algorithm provides an initial efficient allocation of VMs. However, due to dynamic workloads, the resource utilizations of VMs continue to vary over time. Therefore, an initial efficient allocation approach needs to be augmented with a VM consolidation algorithm that can be applied periodically. In our proposed approach, the ACS-VMC algorithm is applied periodically in order to adapt and optimize the VM placement according to the workload.

Fig. 1 depicts the proposed system model that consists of two types of agents: local and global agent. A Local Agent resides in a PM to solve the PM status detection sub-problem by observing the current resource utilizations of the PM. The Global Agent acts as a supervisor and optimizes the VM placement by using the proposed ACS-VMC algorithm. The task sequence of these agents is described as follows:

1) Each LA monitors the CPU utilization and categorizes the PM into one of the four sets P_{normal}, P_{over}, $\hat{P}_{over}$, and P_{under}. Respectively, these sets represent the normal, overloaded, predicted overloaded, and under-loaded PMs based on the following conditions:
   - If the current CPU utilization exceeds PM capacity, the PM is considered as a member of P_{over}.
   - If the predicted utilization value is larger than the available CPU capacity, the PM is considered as a member of $\hat{P}_{over}$. We use LiRCUP [20] to forecast the short-term CPU utilization of a PM based on the linear regression technique. In LiRCUP, the linear regression approximates the utilization function according to the past utilization values in a PM.
   - If the current CPU utilization is less than a threshold of the total CPU utilization, the PM is categorized as a member of P_{under}. We performed a series of preliminary experiments to estimate the threshold. Based on our analysis, in general, the best results are obtained when the threshold is set to 50 percent.
   - All remaining PMs belong to P_{normal}.
2) The GA collects the status of individual PMs from the LAs and builds a global best migration plan by using the proposed ACS-VMC algorithm, which is described in the next section.
3) The GA sends commands to VMMs for performing VM consolidation task. The commands determine

TABLE 1
Summary of Concepts and Their Notations

P	set of physical machines (PMs)
P_{normal}	set of PMs on a normal load level
P_{over}	set of overloaded PMs
$\hat{P}_{over}$	set of predicted overloaded PMs
P_s	set of sleeping PMs
P_{under}	set of under-loaded PMs
V_p	set of VMs running on a PM p
MS	set of migration plans
T	set of tuples
T_k	set of tuples not yet traversed by ant k
V	set of VMs
v	VM in a tuple
$C_{p_{de}}$	total capacity vector of the destination PM p_{de}
M	a migration plan
M^+	the global best migration plan
M_k	ant-specific migration plan of ant k
M_k^m	ant-specific temporary migration plan of ant k
q	a uniformly distributed random variable
S	a random variable selected according to (7)
Scr_k	thus far best score of ant k
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}
p_{de}	destination PM in a tuple
p_{so}	source PM in a tuple
η	heuristic value
τ	amount of pheromone
τ_0	initial pheromone level
$\Delta^+_{\tau_s}$	additional pheromone amount given to the tuples in M^+
q_0	parameter to determine relative importance of exploitation
α	pheromone decay parameter in the global updating rule
β	parameter to determine the relative importance of η
γ	parameter to determine the relative importance of $\|P_s\|$
ρ	pheromone decay parameter in the local updating rule
nA	number of ants that concurrently build their migration plans
nI	number of iterations of the main, outer loop in the algorithm

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.

4 ACS-Based VM Consolidation

The pseudocode of the proposed ACS-based VM Consolidation 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. Each PM $p \in P$ hosts one or more VMs from the set of VMs V. Moreover, in the context of VM migration, each PM is a potential *source PM* for the VMs already residing on that PM. Both the source PM and the VM are characterized by their resource utilizations, such as CPU, memory and network I/O. Likewise, a VM can be migrated to any other PM. Therefore, any other PM is a potential *destination PM*, which is also characterized by its resource utilizations. Thus, the proposed ACS-VMC algorithm creates a set of tuples T, where each tuple $t \in T$ consists of three elements: the source PM p_{so}, the VM to be migrated v, and the destination PM p_{de} as given in (1)

$$t = (p_{so}, v, p_{de}). \tag{1}$$

The PMs in the VM consolidation problem are analogous to the cities in the TSP, while the tuples are analogous to the edges that connect the cities. Due to obvious reasons, it is imperative to reduce the computation time of the consolidation algorithm, which is primarily based on the number of tuples $|T|$. Thus, when making the set of tuples T, the algorithm applies two constraints, which result in a reduced set of tuples by removing some least important and obsolete tuples. The first constraint is given as

$$p_{so} \in \hat{P}_{over} \vee p_{so} \in P_{over} \vee p_{so} \in P_{under}. \tag{2}$$

It ensures that only a predicted overloaded, an overloaded, or an under-loaded PM is used as a source PM p_{so}. The rationale of including a predicted overloaded PM as a source PM is to prevent the PM from becoming overloaded. Similarly, the amount of SLA violations are reduced by migrating some VMs from an overloaded PM. In addition, migrations from an under-loaded PM are more likely to result in switching of the PM to the sleep mode, which would reduce the energy consumption by minimizing the number of active PMs.

The second constraint further restricts the size of the set of tuples $|T|$ by ensuring that none of the overloaded P_{over} and predicted overloaded $\hat{P}_{over}$ PMs become a destination PM p_{de}

$$p_{de} \notin P_{over} \wedge p_{de} \notin \hat{P}_{over}. \tag{3}$$

By applying these two simple constraints in a series of preliminary experiments, we observed that the computation time of the algorithm was significantly reduced without compromising the quality of the solutions.

The output of the VM consolidation algorithm is a migration plan, which, when enforced, would result in a minimal set of active PMs needed to host all VMs without compromising their performance. Thus, the objective function of the proposed algorithm is

$$f(M) = |P_s|^{\gamma} + \frac{1}{|M|}, \tag{4}$$

where M is the migration plan and P_s is the set of PMs that will be switched to the sleep mode when M is enforced. The parameter γ determines the relative importance of $|P_s|$ with respect to $|M|$. Since the ultimate objective in the dynamic VM consolidation algorithm is to minimize the number of active PMs, the objective function is defined in terms of number of sleeping PMs $|P_s|$. Moreover, it prefers smaller migration plans because live migration is a resource-intensive operation.

At the end of the ACS-VMC algorithm, when the selected migration plan is enforced, our approach further restricts the number of active PMs by preferring VM migrations to the already active PMs. Thus, a PM in the sleep mode is switched on only when it is not possible to migrate a VM to an already active PM. Moreover, a PM can only be switched

to the sleep mode when all of its VMs migrate from it, that is, when the PM no longer hosts any VMs. Thus, the set of sleeping PMs P_s is defined as

$$P_s = \{\forall p \in P \mid V_p = \emptyset\}, \tag{5}$$

where V_p is the set of VMs running on a PM p.

Unlike the TSP, there is no notion of a path in the VM consolidation problem. Therefore, in our approach, the pheromone is deposited on the tuples defined in (1). Each of the nA ants uses a stochastic state transition rule to choose the next tuple to traverse. The state transition rule in ACS is called pseudo-random-proportional-rule [11]. According to this rule, an ant k chooses a tuple s to traverse next by applying

$$s = \begin{cases} \arg\max_{u \in T_k}\{[\tau_u] \cdot [\eta_u]^\beta\}, & \text{if } q \leq q_0 \\ S, & \text{otherwise,} \end{cases} \tag{6}$$

where τ denotes the amount of pheromone and η represents the heuristic value associated with a particular tuple. β is a parameter to determine the relative importance of the heuristic value with respect to the pheromone value. $T_k \subset T$ is the set of tuples that remain to be traversed by ant k. $q \in [0,1]$ is a uniformly distributed random variable and $q_0 \in [0,1]$ is a parameter. S is a random variable selected according to the probability distribution given in (7), where the probability p_s of an ant k to choose tuple s to traverse next is defined as

$$p_s = \begin{cases} \frac{[\tau_s]\cdot[\eta_s]^\beta}{\sum_{u \in T_k}[\tau_u]\cdot[\eta_u]^\beta}, & \text{if } s \in T_k, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

The heuristic value η_s of a tuple s is defined in a similar fashion as in [15] and [28] as

$$\eta_s = \begin{cases} \left(|C_{p_{de}} - (U_{p_{de}} + U_v)|_1\right)^{-1}, & \text{if } U_{p_{de}} + U_v \leq C_{p_{de}} \\ 0, & \text{otherwise,} \end{cases} \tag{8}$$

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 in tuple s. The heuristic value η is based on the multiplicative inverse of the scalar-valued difference between $C_{p_{de}}$ and $U_{p_{de}} + U_v$. It favors VM migrations that result in a reduced under-utilization of PMs. Moreover, the constraint $U_{p_{de}} + U_v \leq C_{p_{de}}$ prevents migrations that would result in the overloading of the destination PM p_{de}. In the proposed algorithm, we assumed three resource dimensions, which represent CPU, memory and network I/O utilization.

The stochastic state transition rule in (6) and (7) prefers tuples with a higher pheromone concentration and which result in a higher number of released PMs. The first case in (6) where $q \leq q_0$ is called exploitation [11], which chooses the best tuple that attains the maximum value of $[\tau] \cdot [\eta]^\beta$. The second case, called biased exploration, selects a tuple according to (7). The exploitation helps the ants to quickly converge to a high quality solution, while at the same time, the biased exploration helps them to avoid stagnation by allowing a wider exploration of the search space. In addition to the stochastic state transition rule, ACS also uses a global and a local pheromone trail evaporation rule. The global pheromone trail evaporation rule is applied towards the end of an iteration after all ants complete their migration plans. It is defined as

$$\tau_s = (1 - \alpha) \cdot \tau_s + \alpha \cdot \Delta^+_{\tau_s}, \tag{9}$$

where $\Delta^+_{\tau_s}$ is the additional pheromone amount that is given only to those tuples that belong to the global best migration plan in order to reward them. It is defined as

$$\Delta^+_{\tau_s} = \begin{cases} f(M^+), & \text{if } s \in M^+ \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

$\alpha \in (0,1]$ is the pheromone decay parameter, and M^+ is the global best migration plan from the beginning of the trial.

The local pheromone trail update rule is applied on a tuple when an ant traverses the tuple while making its migration plan. It is defined as

$$\tau_s = (1 - \rho) \cdot \tau_s + \rho \cdot \tau_0, \tag{11}$$

where $\rho \in (0,1]$ is similar to α and τ_0 is the initial pheromone level, which is computed as the multiplicative inverse of the product of the approximate optimal $|M|$ and $|P|$

$$\tau_0 = (|M| \cdot |P|)^{-1}. \tag{12}$$

One way to estimate optimal $|M|$ is to use the nearest neighborhood heuristic [11]. We use the K-nearest neighbor heuristic [30] to estimate the optimal $|M|$ by using a training data set. The data set has m samples, where each sample x_i is described by three input variables (x_{i1}, x_{i2}, x_{i3}) and an output variable y_i, that is, $x_i = \{x_{i1}, x_{i2}, x_{i3}, y_i\}$. The goal is to find a relationship between the input variables and the output variable. Therefore, we choose the number of underloaded PMs, the number of overloaded PMs, and the number of VMs as the three input variables (x_{i1}, x_{i2}, x_{i3}) and the migration plan size as the output variable (y_i). The KNN heuristic estimates the output by taking a local average of the training data set. Moreover, the locality is defined in terms of the K samples nearest to the estimation sample. We use euclidean distance to measure the distance metric between new sample and other samples.

The pseudo-random-proportional-rule in ACS and the global pheromone trail update rule are intended to make the search more directed. The pseudo-random-proportional-rule prefers tuples with a higher pheromone level and a higher heuristic value. Therefore, the ants try to search other high quality solutions in a close proximity of the thus far global best solution. On the other hand, the local pheromone trail update rule complements exploration of other high quality solutions that may exist far from the best-so-far global solution. This is because whenever an ant traverses a tuple and applies the local pheromone trail update rule, the tuple looses some of its pheromone and becomes less attractive for other ants. Therefore, it helps in avoiding stagnation where all ants end up finding the same solution or where they prematurely converge to a suboptimal solution.

The pseudocode in Algorithm 1 creates a set of tuples T using (1) and sets the pheromone value of each tuple to the initial pheromone level τ_0 by using (12) (line 2). The

algorithm iterates over nI iterations (line 3). In each iteration, nA ants concurrently build their migration plans (lines 4-23). Each ant iterates over $|T|$ tuples (lines 76-21). It computes the probability of choosing the next tuple to traverse by using (7) (line 9). Afterwards, based on the computed probabilities and the stochastic state transition rule in (6), each ant chooses a tuple $t \in T_k$ to traverse (line 11) and adds t to its temporary migration plan M_k^m (line 12). The local pheromone trail update rule in (11) and (12) is applied on t (line 13) and the used capacity vectors at the source PM $U_{p_{so}}$ and the destination PM $U_{p_{de}}$ in t are updated to reflect the impact of the migration (line 15). The objective function in (4) is applied on M_k^m, and if it yields a score higher than the ant's best score Scr_k (line 16), t is added to the ant-specific migration plan M_k (line 17). Otherwise, the tuple t is removed from the temporary migration plan M_k^m (line 19). Then, towards the end of an iteration when all ants complete their migration plans, all ant-specific migration plans are added to the set of migration plans MS (line 22). Each migration plan $M_k \in MS$ is evaluated by applying the objective function in (4), the global best application migration plan M^+ is selected (line 24), and the global pheromone trail update rule in (9) and (10) is applied on all tuples (line 25). Finally, when all iterations of the main outer loop complete, the algorithm outputs the global best migration plan M^+.

Algorithm 1. ACS-Based VM Consolidation

1: $M^+ = \emptyset, MS = \emptyset$
2: $\forall t \in T | \tau_t = \tau_0$
3: **for** $i \in [1, nI]$ **do**
4: **for** $k \in [1, nA]$ **do**
5: $M_k^m = \emptyset, M_k = \emptyset, Scr_k = 0$
6: **for** $t \in T$ **do**
7: generate a random variable q with a uniform distribution between 0 and 1
8: **if** $q > q_0$ **then**
9: compute $p_s \; \forall s \in T$ by using (7)
10: **end if**
11: choose a tuple $t \in T_k$ to traverse by using (6)
12: $M_k^m = M_k^m \cup \{t\}$
13: apply local update rule in (11) on t
14: update used capacity vectors $U_{p_{so}}$ and $U_{p_{de}}$ in t
15: **if** $f(M_k^m) > Scr_k$ **then**
16: $Scr_k = f(M_k^m)$
17: $M_k = M_k \cup \{t\}$
18: **else**
19: $M_k^m = M_k^m \setminus \{t\}$
20: **end if**
21: **end for**
22: $MS = MS \cup \{M_k\}$
23: **end for**
24: $M^+ = \arg\max_{M_k \in MS}\{f(M_k)\}$
25: apply global update rule in (9) on all $s \in T$
26: **end for**

5 Experimental Design and Setup

To evaluate the efficiency of our proposed approach, we set up experimental environment using the CloudSim toolkit [13]. CloudSim is a discrete event simulator for implementation and evaluation of resource provisioning and VM consolidation techniques for different applications. We simulated a data center comprising 800 heterogeneous PMs and selected two server configurations in CloudSim: HP ProLiant ML110 G4 (Intel Xeon 3040, 2 cores×1860 MHz, 4 GB), and HP ProLiant ML110 G5 (Intel Xeon 3075, 2 cores×2660 MHz, 4 GB). Dual-core CPUs are sufficient to evaluate resource management methods that are designed for multi-core CPU architectures. Moreover, it is important to simulate a large number of servers for performance evaluation of VM consolidation methods. To evaluate the efficiency of our proposed approach, we measured four metrics: SLA violations, energy consumption, number of migrations, and energy-SLA violations. The results are based on two different workloads: a random workload and a real workload. In the random workload, the users submit requests for the provisioning of 800 heterogeneous VMs. In the real workload, the number of VMs on each day is specified in Table 2. The ACS parameters that were used in the proposed approach are tabulated in Table 3. These parameter values were obtained in a series of preliminary experiments.

TABLE 2
Number of VMs in the Real Workload

Date	Number of VMs
3 March	1,052
6 March	898
9 March	1,061
22 March	1,516
25 March	1,078
3 April	1,463
9 April	1,358
11 April	1,233
12 April	1,054
20 April	1,033

5.1 SLA Violations

Maintaining the desired QoS is an important requirement for cloud data centers. QoS requirements are commonly formalized in the form of SLAs, which specify enterprise service-level requirements for data center in terms of minimum latency or maximum response time. Beloglazov and Buyya [19] proposed a workload independent metric called SLA Violations to evaluate the SLA delivered by a VM in an IaaS cloud. It represents both the SLA Violations due to Over-utilization (SLAVO) and SLA Violations due to Migrations (SLAVM). The SLAVO and SLAVM metrics independently and with equal importance characterize the level of SLA violations by the infrastructure. Therefore, a combined metric (SLAV) describes performance degradations due to the overloading of the host PMs as well as those caused by VM migrations, as

$$SLAV = SLAVO \times SLAVM. \quad (13)$$

SLAVO indicates the percentage of time, during which active PMs have experienced the CPU utilization of 100 percent. It is defined as

TABLE 3
ACS Parameters in the Proposed Approach

α	β	γ	ρ	q_0	nA	nI	w
0.1	0.9	5	0.1	0.9	10	2	2^{-7}

$$SLAVO = \frac{1}{M}\sum_{i=1}^{M}\frac{T_{s_i}}{T_{a_i}}, \tag{14}$$

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

$$SLAVM = \frac{1}{N}\sum_{j=1}^{N}\frac{C_{d_j}}{C_{r_j}}, \tag{15}$$

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. Based on our preliminary experiments, we estimated C_{d_j} as 10 percent of the CPU utilization in MIPS during all migrations of the VM j.

5.2 Energy Consumption

We consider the total energy consumption of the physical resources in a data center that is required to handle the application workloads. The energy consumption of a PM depends on the utilization of its CPU, memory, disk, and network card. Most studies [19], [31] show that CPU consumes more power than memory, disk storage, and network interface. Therefore, the resource utilization of a PM is usually represented by its CPU utilization. Instead of using an analytical model of energy consumption, we used the real data in the SPECpower benchmark.[1] Table 4 illustrates the amount of energy consumption of HP G4 and G5 servers at different load levels.

5.3 Number of VM Migrations

Live VM migration is a costly operation that includes some amount of CPU processing on the source PM, the link bandwidth between the source and destination PMs, the downtime of the services on the migrating VM, and the total migration time [16]. Therefore, one of our objectives was to minimize the number of migrations. The length of a VM migration in CloudSim 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.

5.4 Energy and SLA Violations (ESV)

The objective of the proposed VM consolidation approach is to minimize both energy and SLA violations. Since there is a trade-off between performance and energy consumption, we measured a combined metric called Energy and SLA Violations that captures both the Energy Consumption and the SLA Violations as

$$ESV = EC \times SLAV. \tag{16}$$

6 Experimental Results

In this section, we compare the proposed ACS-VMC approach with an ACS based VM consolidation algorithm in [27] and four heuristic algorithms for dynamic reallocation of VMs in [19]. The AVVMC consolidation scheme [27] proposes the ant colony optimization with balanced usage of computing resources based on vector algebra. Moreover, the main idea of these algorithms [19] is to set upper and lower utilization thresholds and keep the total CPU utilization of a node between them. When the upper threshold is exceeded, VMs are reallocated for load balancing and when the utilization of a PM drops below the lower threshold, VMs are reallocated for consolidation. The algorithms adapt the utilization threshold dynamically based on the Median Absolute Deviation (MAD), the Interquartile Range (IQR), and Local Regression (LR) approach to estimate the CPU utilization. Moreover, a static threshold method (THR) is proposed in [19] that monitors the CPU utilization and migrates a VM when the current utilization exceeds 80 percent of the total amount of available CPU capacity on the PM. In our experiments, we consider two type of workloads:

6.1 Random Workload

In the random workload, each VM runs an application with a variable utilization of CPU, which is generated with a uniform distribution. Fig. 2a presents the SLA violation levels caused by the ACS-VMC, AVVMC, THR, MAD, IQR, and LR methods in the random workload. The results indicate that ACS-VMC reduced the SLA violations more efficiently than the other approaches. This is due to the fact that ACS-VMC prevents SLA violations by using a prediction of the overloaded PMs and that the heuristic value in (8) ensures that the destination PM does not become overloaded when a VM migrates on it. Fig. 2b shows that the proposed dynamic VM consolidation approach, ACS-VMC, brought higher energy savings in comparison to the other approaches in the random workload. In ACS-VMC, a signicant reduction of the energy consumption of 7.3, 16.4, 45.2, 34.5, and 47.7 percent was achieved when compared to AVVMC, LR, MAD, THR, and IQR, respectively. In addition, the trade-off between maximizing the QoS and minimizing the energy consumption of data center is demonstrated in Fig. 2c. Fig. 2d depicts the total number of VMs migration during the VM consolidation in the random workload. The ACS-VMC outperforms the AVVMC and adaptive-threshold based algorithms due to predictions of utilization, and therefore decreased the number of VM migrations.

6.2 Real Workload

Real workload data is provided as a part of the CoMon project, a monitoring infrastructure for PlanetLab [32]. In this project, the CPU usage data is collected every five

1. http://www.spec.org/power_ssj2008/

TABLE 4
Energy Consumption at Different Load Levels in Watts

Server	sleep mode	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

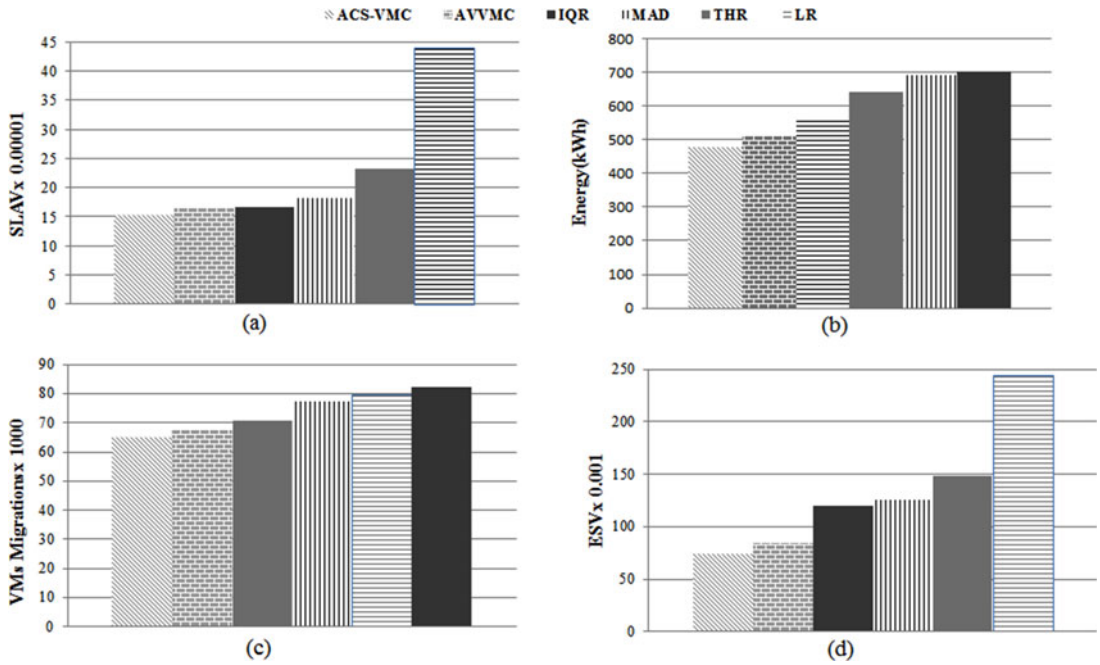


Fig. 2. The SLAV metric, energy consumption, number of VM migrations and ESV metric by ACS-VMC and benchmark methods in the random workload.

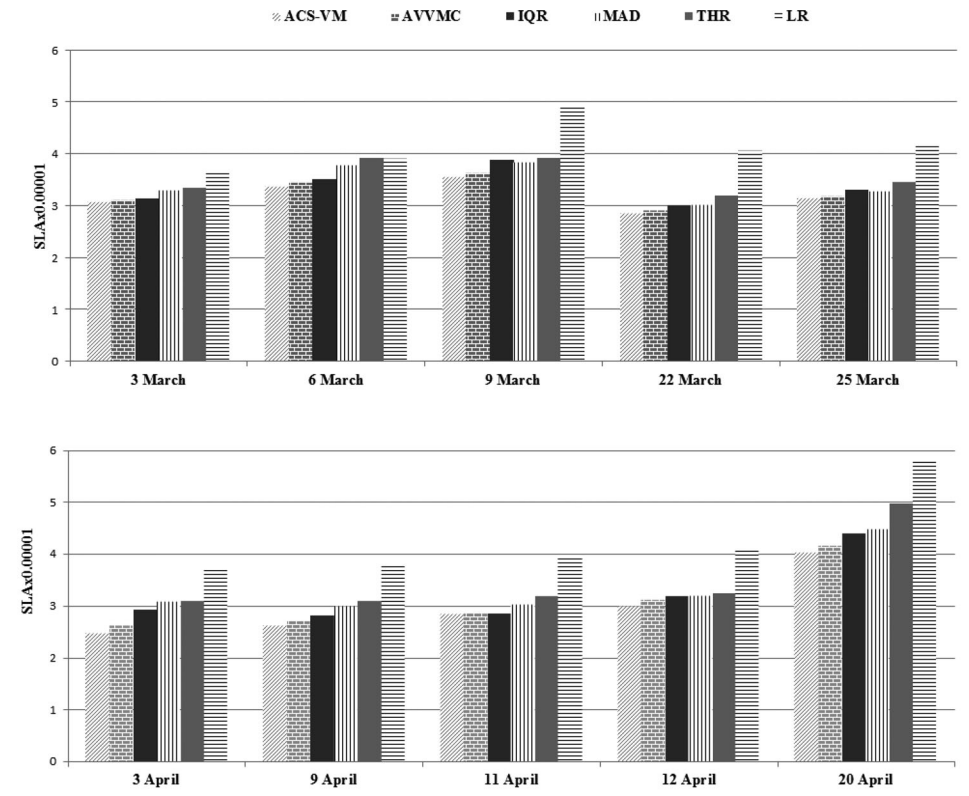
minutes from more than a thousand VMs and is stored in different files. The VMs are allocated on servers that are located at more than 500 places around the world. In fact, the workload is representative of an IaaS cloud environment such as Amazon EC2, which several independent users create and manage VMs. Fig. 3a shows that ACS-VMC led to significantly less SLA violations than the other four benchmark algorithms. The main reason is that ACS-VMC employs measures to prevent VM migrations that would result in the overloading of the destination PM. Moreover, it preemptively reallocates VMs from a predicted overloaded PM. Fig. 3b shows that ACS-VMC consumed less power than the other benchmark algorithms in the real workload traces. our proposed VM consolidation approach reduces energy consumption by up to 53.4 percent with desirable system performance in March 2011 load traces. This is because, the defined objective function tries to maximize the number of dormant PMs by packing VMs into the PMs that have enough capacity. The number of VM migrations in the real workload are shown in Fig. 4a. As observed from the results, ACS-VMC has minimum number of migrations compared with the other benchmark methods. Because it creates a migration plan that has require the minimum number of migrations. In addition, Fig. 4b illustrates the ACS-VMC consumes less ESV than other benchmarks algorithms in the real workload traces.

7 Conclusion and Future Work

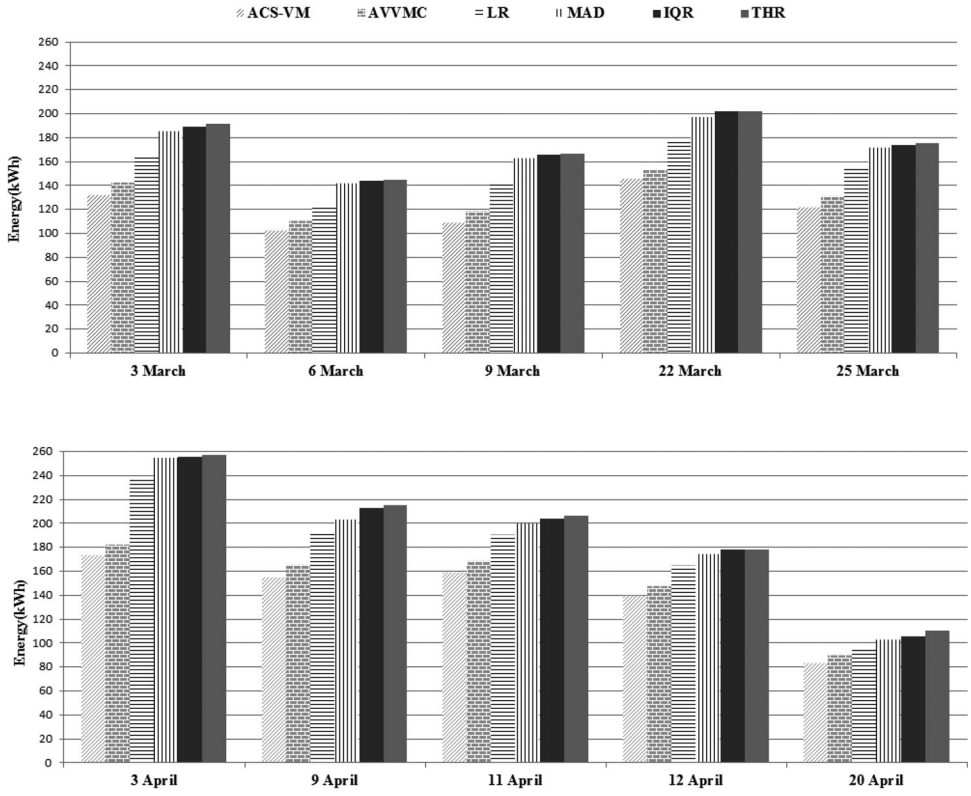
In this paper, we presented a novel dynamic Virtual Machine consolidation approach called ACS-based VM Consolidation. It reduces the energy consumption of data centers by consolidating VMs into a reduced number of active Physical Machines while preserving Quality of Service requirements. Since the VM consolidation problem is strictly NP-hard, we used the Ant Colony System to find a near-optimal solution. We defined a multi-objective function that considers both the number of dormant PMs and the number of migrations. When compared to the existing dynamic VM consolidation approaches, ACS-VMC not only reduced the energy consumption, but also minimized SLA violations and the number of migrations. We evaluated the performance of our proposed approach by conducting experiments with ten different real workload traces.

As a future work, we plan to further improve the proposed system model by clustering PMs and assigning them to the respective consolidation managers. We also intend to evaluate the performance of other heuristic methods for VM consolidation. Furthermore, we plan to implement the ACS-VMC algorithm as an extension of the VM manager within the OpenStack Cloud platform[2] to evaluate the proposed VM consolidation algorithm in a real cloud environment.

2. http://openstack.org/

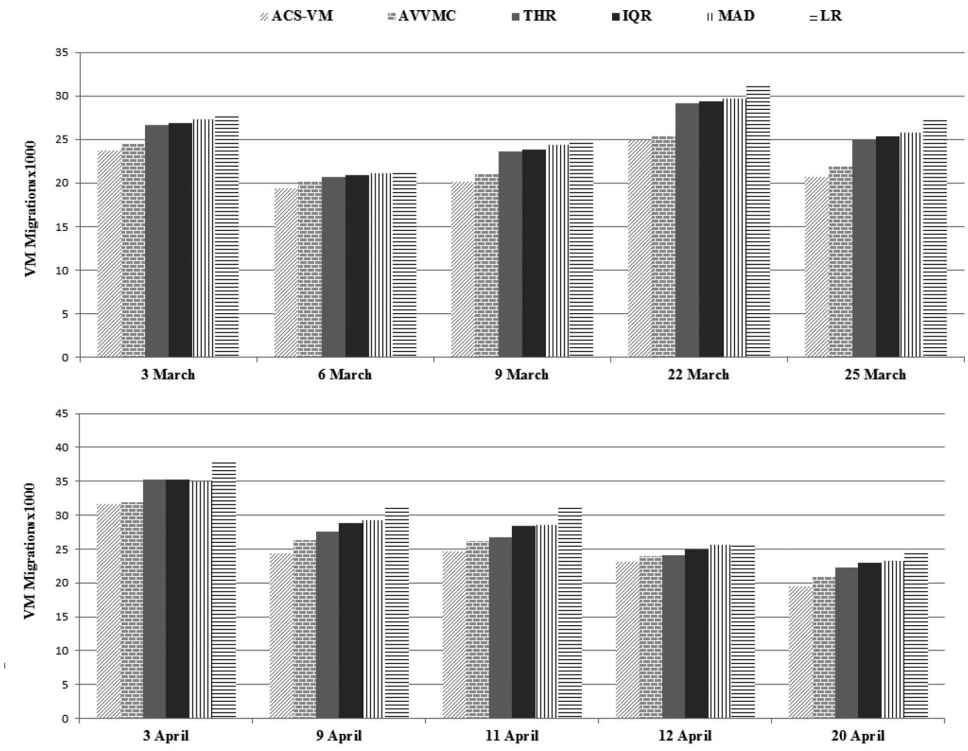


(a) SLAV metric

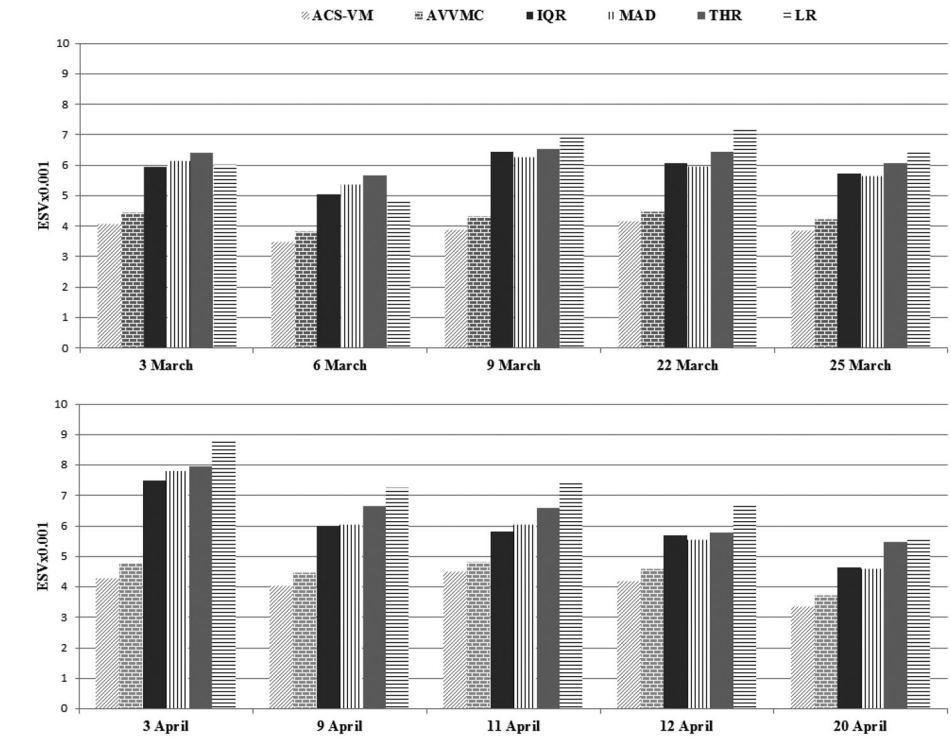


(b) Energy consumption

Fig. 3. SLAV metric and energy consumption by ACS-VMC and benchmark methods in the real workloads.



(a) Number of VM migrations



(b) ESV metric

Fig. 4. Number of VM migrations and ESV metric by ACS-VMC and benchmark methods in the real workloads.

REFERENCES

[1] P. Mell and T. Grance. (2011, Sept.). The NIST definition of cloud computing Recommendations of the National Institute of Standards and Technology. Special Publication 800-145 [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[2] G. Motta, N. Sfondrini, and D. Sacco, "Cloud computing: An architectural and technological overview," in *Proc. Int. Joint Conf. Serv. Sci.*, 2012, pp. 23–27.

[3] I. Sriram and A. Khajeh-Hosseini, "Research agenda in cloud technologies," Large Scale Complex IT Syst. (LSCITS), Univ. Bristol, U.K., 2010, http://arxiv.org/ftp/arxiv/papers/1001/1001.3257.pdf

[4] A. Ashraf, "Cost-efficient virtual machine management: Provisioning, admission control, and consolidation," Ph.D. dissertation, Turku Centre for Computer Science (TUCS) Dissertations Number 183, Åbo, Finland, Oct. 2014.
[5] A. Ashraf, M. Hartikainen, U. Hassan, K. Heljanko, J. Lilius, T. Mikkonen, I. Porres, M. Syeed, and S. Tarkoma, "Introduction to cloud computing technologies," in *Developing Cloud Software: Algorithms, Applications, and Tools*, I. Porres, T. Mikkonen, and A. Ashraf, Eds. Åbo, Finland: Turku Centre for Computer Science (TUCS) General Publication Number 60, Oct. 2013, pp. 1–41.
[6] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *Adv. Comput.*, vol. 82, pp. 47–111, 2011.
[7] A. Gandhi, M. Harchol-Balter, R. Raghunathan, and M. A. Kozuch, "AutoScale: Dynamic, robust capacity management for multi-tier data centers," *ACM Trans. Comput. Syst.*, vol. 30, no. 4, pp. 14:1–14:26, 2012.
[8] L. Deboosere, B. Vankeirsbilck, P. Simoens, F. Turck, B. Dhoedt, and P. Demeester, "Efficient resource management for virtual desktop cloud computing," *J. Supercomput.*, vol. 62, no. 2, pp. 741–767, 2012.
[9] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, 2003.
[10] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proc. 2nd Conf. Symp. Netw. Syst. Design Implementation*, 2005, vol. 2, pp. 273–286.
[11] M. Dorigo and L. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evolutionary Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.
[12] M. Dorigo, G. Di Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artif. Life*, vol. 5, no. 2, pp. 137–172, Apr. 1999.
[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," *Softw.: Prac. Exp.*, vol. 41, no. 1, pp. 23–50, 2011.
[14] W. Vogels, "Beyond server consolidation," *ACM Queue*, vol. 6, no. 1, pp. 20–26, 2008.
[15] E. Feller, C. Morin, and A. Esnault, "A case for fully decentralized dynamic VM consolidation in clouds," in *Proc. IEEE 4th Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2012, pp. 26–33.
[16] A. Murtazaev and S. Oh, "Sercon: Server consolidation algorithm using live migration of virtual machines for green computing," *IETE Tech. Rev.*, vol. 28, no. 3, pp. 212–231, 2011.
[17] M. Marzolla, O. Babaoglu, and F. Panzieri, "Server consolidation in clouds through gossiping," in *Proc. IEEE Int. Symp. World Wireless, Mobile Multimedia Netw.*, 2011, pp. 1–6.
[18] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Comput. Syst.*, vol. 28, pp. 755–768, 2012.
[19] 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 Comput.: Prac. Exp.*, vol. 24, no. 13, pp. 1397–1420, 2012.
[20] 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 *Proc. 39th EUROMICRO Conf. Softw. Eng. Adv. Appl.*, 2013, pp. 357–364.
[21] F. Farahnakian, T. Pahikkala, P. Liljeberg, and J. Plosila, "Energy aware consolidation algorithm based on K-nearest neighbor regression for cloud data centers," in *Proc. IEEE/ACM 6th Int. Conf. Utility Cloud Comput.*, Dec. 2013, pp. 256–259.
[22] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Comput. Netw.*, vol. 53, pp. 2923–2938, 2009.
[23] Y. Ajiro and A. Tanaka, "Improving packing algorithms for server consolidation," in *Proc. Int. Conf. Comput. Meas. Group*, 2007, pp. 399–407.
[24] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *Proc. IEEE Conf. Comput. Commun.*, 2011, pp. 71–75.
[25] M. Harman, K. Lakhotia, J. Singer, D. R. White, and S. Yoo, "Cloud engineering is search based software engineering too," *J. Syst. Softw.*, vol. 86, no. 9, pp. 2225–2241, 2013.
[26] P.-Y. Yin and J.-Y. Wang, "Ant colony optimization for the nonlinear resource allocation problem," *Appl. Math. Comput.*, vol. 174, no. 2, pp. 1438–1453, 2006.
[27] M. Ferdaus, M. Murshed, R. Calheiros, and R. Buyya, "Virtual machine consolidation in cloud data centers using ACO metaheuristic," in *Proc. 20th Int. Eur. Conf. Parallel Process.*, 2014, vol. 8632, pp. 306–317.
[28] A. Ashraf and I. Porres, "Using ant colony system to consolidate multiple web applications in a cloud environment," in *Proc. 22nd Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process.*, 2014, pp. 482–489.
[29] M. Mishra and A. Sahoo, "On theory of VM placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach," in *Proc. IEEE Int. Conf. Cloud Comput.*, Jul. 2011, pp. 275–282.
[30] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.
[31] D. Kusic, J. Kephart, J. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," in *Proc. Int. Conf. Auton. Comput.*, 2008, pp. 3–12.
[32] K. Park and V. Pai, "CoMon: A mostly-scalable monitoring system for PlanetLab," *ACM SIGOPS Oper. Syst. Rev.*, vol. 40, pp. 65–74, 2006.

Fahimeh Farahnakian received the MS degree in computer engineering from the University of Science and Technology, Tehran, Iran, in 2009. She is currently working toward the PhD degree in the Embedded Computer and Electronic Systems Laboratory, University of Turku, Finland. Since May 2011, she has been a doctoral candidate of Graduate School in Electronics, Telecommunications and Automation (GETA). In addition, her research interests include cloud computing, energy efficient data center, network on-chips, machine learning, pattern recognition and fuzzy control. She 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*. She is a member of the IEEE.

Adnan Ashraf received the MSc and MS degrees in computer science from Mohammad Ali Jinnah University, Islamabad, Pakistan, in 2003 and 2006, respectively, and the PhD degree in software engineering from Åbo Akademi University, Turku, Finland, in 2014. He is currently working as a postdoctoral researcher in the Software Engineering Laboratory at Åbo Akademi University. His research interests include cloud computing, energy-efficient data centers, cost-efficient virtual machine management, admission control, server consolidation, and search-based software engineering.

Tapio Pahikkala received the PhD degree in computer science from the University of Turku in 2008 and held an Academy of Finland postdoctoral position during 2010-2012. He currently acts as a professor of intelligence systems in the 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 the ACM and the IEEE, and he served as member in several committees of IEEE societies.

Pasi Liljeberg received the MSc and PhD 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. During the period 2007-2009, he held an Academy of Finland researcher position. He is the author of more than 150 peer-reviewed publications, has supervised nine PhD theses. His current research interests include parallel and distributed systems, Internet-of-Things, embedded computing architecture, fault tolerant and energy aware system design, 3D multiprocessor system architectures, dynamic power management, cyber physical systems, intelligent network-on-chip communication architectures and reconfigurable system design.

Juha Plosila received the PhD degree in electronics and communication technology from University of Turku (UTU) in 1999. He 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. He is the leader of the Embedded Computer and Electronic Systems (ECES) research unit and a co-leader of the Resilient IT Infrastructures (RITES) research programmer at Turku Centre for Computer Science (TUCS). He leads the Embedded Systems Masters Program at the EIT ICT Labs Master School and is a management committee member of the EU COST Actions IC1103 (MEDIAN: Manufacturable and Dependable Multicore Architectures at Nanoscale) and IC1202 (TACLe: Timing Analysis on Code Level). He is an associate editor of *International Journal of Embedded and Real-Time Communication Systems (IGI Global)*.

Ivan Porres is a professor in software engineering, the head of the computer engineering education, and the vice-head of the Department of Information Technologies at Åbo Akademi University. He is the leader of the Software Engineering Laboratory at the Turku Centre for Computer Science (TUCS) and principal investigator at Åbo Akademi for the Cloud Software Finland (2009-2013) and N4S (2014-2017) projects at the DIGILE, the Finnish Strategic Centre for Science, Technology and Innovation in the Field of ICT. He has received the 10-Year Most Influential Paper Award at the ACM/IEEE Conference on Model Driven Engineering Languages and Systems in two occasions and has participated in many review appointments and the organization of research events.

Hannu Tenhunen received the diplomas from the Helsinki University of Technology, Finland, 1982, and the PhD degree from Cornell University, Ithaca, NY, 1986. In 1985, he joined the Signal Processing Laboratory, Tampere University of Technology, Finland, as an associate professor and later served as a professor and department director. Since 1992, ha has been a professor at the Royal Institute of Technology (KTH), Sweden, where he also served as a 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.