Energy-Aware Dynamic VM Consolidation in Cloud Data Centers Using Ant Colony System

Fahimeh Farahnakian*, Adnan Ashraf†‡, Pasi Liljeberg*, Tapio Pahikkala*, Juha Plosila*, Ivan Porres†, Hannu Tenhunen*
* Department of Information Technology, University of Turku, Turku, Finland.
† Department of Information Technologies, Åbo Akademi University, Turku, Finland.
‡ Department of Software Engineering, International Islamic University, Islamabad, Pakistan.
fahfar@utu.fi, aashraf@abo.fi, pakrli@utu.fi, aatapa@utu.fi, juplos@utu.fi, iporres@abo.fi, hanten@utu.fi

***Abstract*—As the scale of a cloud data center becomes larger and larger, the energy consumption of the data center also grows rapidly. Dynamic consolidation of Virtual Machines (VMs) presents a significant opportunity to save energy by turning off unused Physical Machines (PMs) in data centers. In this paper, we present a distributed controller to perform dynamic VM consolidation to improve the resource utilizations of PMs and to reduce their energy consumption. Moreover, we use the ant colony system to find a near-optimal VM placement solution based on the specified objective function. Experimental results on the real workload traces from more than a thousand PlanetLab VMs show that the proposed approach reduces energy consumption and maintains required performance levels in a large-scale data center.**

***Keywords*-Dynamic VM consolidation; ant colony system; cloud computing; green computing; energy-efficiency; SLA**

I. Introduction

Several major Information and Communication Technology (ICT) companies, such as Amazon, Google, Microsoft, and Facebook, are operating large-scale data centers around the world to handle the ever-increasing cloud infrastructure demand. However, the growing demand has considerably increased the energy consumption of the cloud data centers [1]. High energy consumption not only translates to a higher cost, but also leads to higher carbon emissions. Therefore, energy-related costs have become a major economical factor for data centers and research communities are being challenged to find efficient energy-aware resource management strategies. Moreover, achieving the desired level of Quality of Service (QoS) between cloud providers and their customers is critical in a data center. The QoS requirements are formalized via Service Level Agreements (SLAs) that describe the required characteristics, such as minimal throughput and maximal response time or latency of the system.

Dynamic server provisioning and Virtual Machine (VM) consolidation are two high-level approaches to reduce power consumption in data centers. Dynamic server provisioning approaches [2] save power by powering only a minimum amount of resources needed to satisfy the workload requirements. Therefore, unnecessary servers are brought offline or put into a low-power mode if the workload demand decreases. Similarly, when the demands increase, additional servers are turned online. Dynamic VM consolidation is another effective way to improve the utilization of resources and their energy efficacy [1], [3]. It leverages virtualization technology [4], 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), which resides in a PM. 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 minimum number of PMs, reducing the power consumption [5]. However, resources must be properly allocated and load balancing must be guaranteed in order to maximize resource utility. Therefore, how to efficiently place VMs is an important research problem in the VM consolidation approaches. Furthermore, the VM placement optimization should be performed in an online manner to cope with the workload variability of different applications.

In this paper, we present a distributed controller to perform dynamic VM consolidation to improve the resource utilizations of PMs and to reduce their energy consumption. We also propose a dynamic VM consolidation algorithm that uses a highly adaptive online optimization metaheuristic called Ant Colony Optimization (ACO) to optimize VM placement. The proposed approach uses artificial ants to consolidate VMs into a minimum number of active PMs according to the current resource requirements. These ants work in parallel to build VM migration plans based on the specified objective function. The performance of the proposed dynamic VM consolidation approach is evaluated by using CloudSim [6] 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 the proposed approach maintains the desired QoS while reducing energy consumption in a large-scale data center. In contrast to the existing benchmark algorithms in the CloudSim toolkit, it provides a more energy-efficient solution with fewer SLA violations.

The remainder of this paper is organized as follows. Section II discusses some of the most important related works and briefly reviews the ACO metaheuristic. Section III and Section IV present the system architecture and the proposed

dynamic VM consolidation approach, respectively. Section V describes the experimental design and setup. Finally, we present the experimental results in Section VI and our conclusions in Section VII.

II. Background and Related Work

The existing VM consolidation approaches, such as [7], [8], [9], [10] are used in data centers to minimize under-utilization of PMs and optimize their power-efficiency. The main idea in these approaches is to use live VM migration [5] to periodically consolidate VMs so that some of the under-loaded PMs could 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 [11], two static thresholds were used to indicate the time of VM reallocation. This approach keeps the total CPU utilization of a PM between these thresholds. However, setting static thresholds is not efficient for an environment with dynamic workloads, in which different types of applications may run on a PM. Therefore, Beloglazov and Buyya [12] improved the idea by considering adaptive upper and lower bounds based on the statistical analysis of the historical data.

In our previous works [13], [14], we proposed two regression methods to predict CPU utilization of a PM. These methods use the linear regression and the K-nearest neighbor regression algorithms 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 power consumption.

In some approaches, VM consolidation has been formulated as an optimization problem [15], [16], [17]. 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 bins and the VMs are objects. These algorithms solve this problem to minimize the number of bins while packing all the objects. In this paper, we formulate energy-efficient VM consolidation as a combinatorial optimization problem and apply a highly adaptive online optimization [18] metaheuristic called Ant Colony Optimization (ACO) [19], [20] 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 [19]. It is inspired by the foraging behavior of real ant colonies. While moving from their nest to the 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 [19]. 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 [19]. 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 (ACS) [19], [20]. ACS [20] was introduced to improve the performance of AS and it is currently one of the best performing ant algorithms. Therefore, in this paper, we apply ACS to the VM consolidation problem.

One of the earlier works on applying ACO to the general resource allocation problem include [21]. The authors in [21] 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. [8] applied MMAS to the VM consolidation problem in the context of cloud computing. A more recent work by Ashraf and Porres [22] used ACS to consolidate multiple web applications in a cloud-based shared hosting environment. However, to the best of our knowledge, currently there are no existing works on using ACS to consolidate VMs in cloud data centers. Our main contributions are as follows:

- We propose a distributed controller for performing the VM consolidation task. The key advantage of the proposed distributed controller is that it splits the complex and large consolidation problem into two smaller sub-problems: PM status detection and VM placement optimization. The first sub-problem concerning PM status is addressed by a local controller that resides in a PM, while the second sub-problem of VM placement optimization is solved by a global controller.
- The first sub-problem concerns PM status detection: normal, overloaded, predicted overloaded, or under-loaded. We use the LiRCUP method [13] to predict an overloaded PM for avoiding SLA violations, as described in Section III.
- The second sub-problem of VM placement optimization is a NP-hard problem. In our proposed approach, the global controller efficiently solves this problem by using the ACS-based Placement Optimization (ACS-PO) algorithm, which is presented in Section IV.
- We use the K-nearest neighbor (KNN) heuristic [23] to estimate the optimal migration plan size for calculating the amount of initial pheromone in ACS.

III. System Architecture

We consider a large-scale data center as a resource provider that consists of m heterogeneous PMs. Each PM has a processor, which can be multi-core, with performance defined in Millions of Instructions Per Second (MIPS). Besides that,

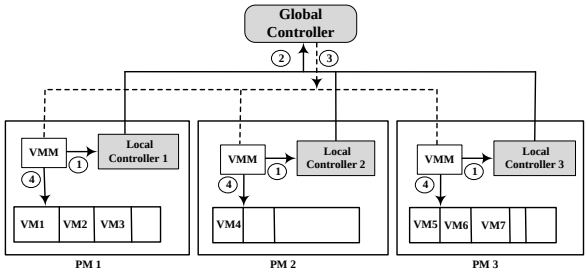

Fig. 1. An example of the system architecture

a PM is characterized by the amount of RAM, network bandwidth, and storage capacity. Several users submit requests for provisioning of n VMs, which are allocated to the PMs. The length of each request is specified in millions of instructions (MI). Initially, the VMs are allocated according to the requested characteristics assuming 100% CPU utilization. Afterwards, the proposed dynamic consolidation algorithm optimizes VM placement to reduce the power consumption and SLA violations of the data center.

Figure 1 depicts an example system model with three PMs and seven VMs. Our proposed distributed architecture divides the large VM consolidation problem into two smaller sub-problems. These sub-problems are addressed by two controllers: the global controller and the local controller. A local controller is assumed for each PM to solve the PM status detection sub-problem by observing the current utilization. Moreover, the architecture has a global controller as a supervisor to optimize the VM placement sub-problem by using our proposed ACS-PO algorithm. The task sequence of the controllers is described as follows:

1) A local controller resides on a PM. It monitors the CPU utilization and classifies 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 [13] 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 assumed 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 were obtained when the threshold was set to 50%.
   - All remaining PMs belong to P_{normal}.
2) The global controller collects the status of individual PMs from the local controllers and builds a global best migration plan by using the proposed ACS-PO algorithm, which is described in the next section.
3) The global controller sends commands to VMMs for VM placement optimization. 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 global controller.

IV. ACS-Based VM Placement Optimization

The pseudocode of the proposed ACS-based Placement Optimization (ACS-PO) algorithm is given as Algorithm 1. For the sake of clarity, the concepts used in the ACS-PO algorithm and their notations are tabulated in Table I. 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 and memory utilization. Likewise, a VM can be migrated to any other PM. Therefore, every other PM is a potential *destination PM*, which is also characterized by its resource utilizations. Thus, the proposed ACS-PO 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 unwanted 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}

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

The rationale is that migrations to an overloaded or a predicted overloaded PM are more likely to put additional load on the destination PM and to cause SLA violations. By applying these

TABLE I
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_{sleep}	set of sleep 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 $\lvert P_{sleep}\rvert$
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

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

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

where M is the migration plan and P_{sleep} is the set of PMs that will be switched to sleep mode when M is enforced. The parameter γ determines the relative importance of $|P_{sleep}|$ 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 sleep PMs $|P_{sleep}|$. Moreover, it prefers smaller migration plans because live migration is a resource-intensive operation. Thus, the objective function in (4) prefers a migration plan that results in the minimum number of active PMs and that requires fewer VM migrations.

At the end of the ACS-PO 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 sleep PMs P_{sleep} is defined as

$$P_{sleep} = \{\forall p \ \in \ P \mid V_p = \emptyset\} \tag{5}$$

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 [20]. 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. The expression *arg max* returns the tuple for which $[\tau] \cdot [\eta]^{\beta}$ attains its maximum 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 [8] and [22] as

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

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 two resource dimensions, which represent CPU and memory utilization. However, if necessary, it is possible to add more dimensions in the total and used capacity vectors.

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 [20], 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 [20]. We use the K-nearest neighbor (KNN) heuristic [23] 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 the relationship between the input variables and the output variable. Therefore, we choose the number of under-loaded 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 quarry 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 thus far global best 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 thus 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.

Algorithm 1 ACS-based VM placement optimization

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: $M^m_k = \emptyset, M_k = \emptyset, Scr_k = 0$
6: **for** $t \in T$ **do**
7: **if** $q > q_0$ **then**
8: compute $p_s \quad \forall s \in T$ by using (7)
9: **end if**
10: choose a tuple $t \in T_k$ to traverse by using (6)
11: $M^m_k = M^m_k \cup \{t\}$
12: apply local update rule in (11) on t
13: update used capacity vectors $U_{p_{so}}$ and $U_{p_{de}}$ in t
14: **if** $f(M^m_k) > Scr_k$ **then**
15: $Scr_k = f(M^m_k)$
16: $M_k = M_k \cup \{t\}$
17: **else**
18: $M^m_k = M^m_k \setminus \{t\}$
19: **end if**
20: **end for**
21: $MS = MS \cup \{M_k\}$
22: **end for**
23: $M^{+} = \arg\max_{M_k \in MS}\{f(M_k)\}$
24: apply global update rule in (9) on all $s \in T$
25: **end for**

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 4–22). Each ant iterates over $|T|$ tuples (lines 6–20). It computes the probability of choosing the next tuple to traverse by using (7) (line 8). 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 10) and adds t to its temporary migration plan M^m_k (line 11). The local pheromone trail update rule in (11) and (12) is applied on t (line 12) 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 13). The objective function in (4) is applied on M^m_k, and if it yields a score higher than the ant's thus far best score Scr_k (line 14), t is added to the ant-specific migration plan M_k (line 16). Otherwise, the tuple t is removed from the temporary migration plan M^m_k (line 18). 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 21). Each migration plan $M_k \in MS$ is evaluated by applying the objective function in (4), the thus far global best application migration plan M^+ is selected (line 23), and the global pheromone trail update rule in (9) and (10) is applied on all tuples (line 24). Finally, when all iterations of the main, outer loop complete, the algorithm outputs the global best migration plan M^+.

V. Experimental Design and Setup

To evaluate the efficiency of our proposed approach, we have developed software simulations by using the CloudSim toolkit [6]. CloudSim is becoming increasingly popular in the cloud computing community due to its support for flexible, scalable, efficient, and repeatable evaluations of provisioning policies 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). The reason why we have not chosen servers with more cores is that it is important to simulate a large number of servers to evaluate the effect of consolidation. Nevertheless, dual-core CPUs are sufficient to evaluate resource management algorithms designed for multi-core CPU architectures. The frequency of the servers CPUs are mapped onto MIPS ratings: 1860 MIPS each core of the HP ProLiant ML110 G4 server, and 2660 MIPS each core of the HP ProLiant ML110 G5 server. Each server is modeled to have 1 GB/s network bandwidth.

In our experiments, we considered a random workload and a real workload. In the random workload, users submit their requests for provisioning of 800 heterogeneous VMs that fill the full capacity of the simulated data center. Each VM runs an application with a variable workload, which was designed to generate CPU utilization according to a uniformly distributed random variable. The application runs for 150000 MI that is equal to 10 minutes of execution on a 250 MIPS CPU core with 100% utilization.

In the real workload, the number of VMs on each day is specified in Table II. Real workload data is provided as a part of the CoMon project, a monitoring infrastructure for PlanetLab [24]. In this project, the CPU utilization data is obtained from more than a thousand VMs from servers located at more than 500 places around the world. Data is collected every five minutes and is stored in a variety of files. The workload is representative of an IaaS cloud environment, such as Amazon Elastic Compute Cloud (EC2)[1], where VMs are created and managed by several independent users and the infrastructure provider is not aware of the particular applications that run in the VMs. We used 10 days from the workload traces collected during 2011. During the simulation, each VM was randomly assigned a workload trace from one of the VMs from the corresponding day. The ACS parameters that were used in the proposed approach are tabulated in Table III. These parameter values were obtained in a series of preliminary experiments.

[1]http://aws.amazon.com/ec2/

TABLE II
Number of VMs in the real workload

Date	Number of VMs
3 March	1052
6 March	898
9 March	1061
22 March	1516
25 March	1078
3 April	1463
9 April	1358
11 April	1233
12 April	1054
20 April	1033

TABLE III
ACS parameters in the proposed approach

α	β	γ	ρ	q_0	nA	nI
0.1	0.9	5	0.1	0.9	10	2

VI. Experimental Results

In this section, we compare the proposed ACS-PO approach with the three heuristic algorithms for dynamic reallocation of VMs in [12]. The main idea of these algorithms 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. In addition, we consider the static threshold method (THR) in [12] that monitors the CPU utilization and migrates a VM when the current utilization exceeds 80% of the total amount of available CPU capacity on the PM. The comparison is based on two performance metrics: average SLA violation percentage and energy consumption.

A. Average SLA Violation Percentage

This metric represents the percentage of average CPU performance that has not been allocated to an application when requested, resulting in a performance degradation [11]. It is calculated by (13) as a fraction of the difference between the MIPS requested by all VMs $U_{rj}(t)$ and the actually allocated MIPS $U_{aj}(t)$ relative to the total requested MIPS over the lifetime of the VMs, where M is the number of VMs.

$$SLA = \frac{\sum_{j=1}^{M} \int U_{rj}(t) - U_{aj}(t) dt}{\sum_{j=1}^{M} \int U_{rj}(t) dt} \tag{13}$$

Table IV presents the SLA violation levels caused by the ACS-PO, THR, MAD, IQR, and LR methods in the random workload. The results indicate that ACS-PO reduced the

TABLE IV
AVERAGE SLA VIOLATION PERCENTAGE IN THE RANDOM WORKLOAD

ACS-PO (%)	THR (%)	MAD (%)	IQR (%)	LR (%)
7.98	12.75	10.75	10.35	14.89

TABLE V
AVERAGE SLA VIOLATION PERCENTAGE IN THE REAL WORKLOAD

Date	ACS-PO (%)	THR (%)	MAD (%)	IQR (%)	LR (%)
3 March	4.78	10.14	10.18	9.98	6.90
6 March	8.36	10.13	10.05	10.17	9.63
9 March	8.84	10.25	10.35	10.14	10.11
22 March	8.13	10.25	10.16	10.19	9.58
25 March	8.21	10.11	9.96	10.09	9.64
3 April	7.23	10.07	10.11	10.05	10.05
9 April	8.46	10.25	10.05	10.01	10.16
11 April	8.78	10.08	10.10	10.01	10.41
12 April	8.81	10.27	10.04	10.17	10.45
20 April	9.10	10.75	10.49	10.35	11.28

average SLA violation percentage more efficiently than the other approaches. This is due to the fact that ACS-PO 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.

The percentages of average SLA violation for the real workload are shown in Table V. The results show that ACS-PO led to significantly less SLA violations than the other four benchmark algorithms. The main reason is that ACS-PO 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.

B. Energy Consumption

Since one of the essential objectives of dynamic VM consolidation is to minimize the energy consumption of the PMs, it is an important performance metric for the comparison of the algorithms. This metric indicates the total energy consumption by the physical resources of a data center caused by the application workloads. Table VI illustrates the power consumption characteristics of the selected servers in the simulator.

Since the CPU utilization usually changes over time due to workload variability, it is defined as a function of time, represented as $u(t)$. The total energy consumption E of a PM can be defined as an integral of the power consumption function over a period of time as

$$E = \int_{t_0}^{t_1} P(u(t))dt \tag{14}$$

Figure 2 shows that the proposed dynamic VM consolidation approach ACS-PO brought higher energy savings in comparison to the other approaches in the random workload. In ACS-PO, a signicant reduction of the energy consumption of 10.1%, 37.4%, 27.2%, and 39.8% was achieved when compared to LR, MAD, THR, and IQR, respectively.

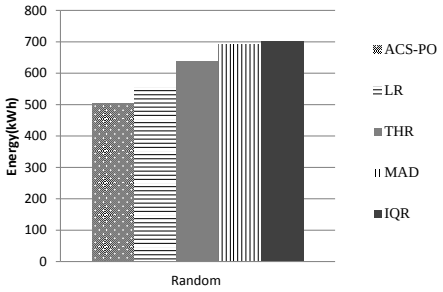

Fig. 2. Energy consumption by ACS-PO and benchmark methods in the random workload

Figure 3 and 4 show that ACS-PO consumed less power than the other benchmark algorithms in the real workload traces. Figure 3 depicts that it reduced energy consumption by up to 21.9% with desirable system performance in March 2011 load traces. Moreover, it achieved a signicant reduction of 6.1%, 15.7%, 12.9%, and 21.3% in the energy consumption when compared to LR, THR, MAD, and IQR methods in April 2011 traces, respectively (Figure 4).

VII. CONCLUSION

In this paper, we presented a novel dynamic Virtual Machine (VM) consolidation approach to reduce the energy consumption of data centers by packing VMs into a minimum number of active Physical Machines (PMs) according to the current resource requirements. It employs a distributed controller to decompose the VM consolidation problem into two sub-problems: PM status detection and VM placement optimization. Since the VM placement optimization problem is NP-hard, we used the ant colony system to find a near-optimal solution. When compared to the existing dynamic VM consolidation approaches in the CloudSim toolkit, the proposed approach provided a more energy-efficient solution with fewer SLA violations.

REFERENCES

[1] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *Advances in Computers*, vol. 82, pp. 47–111, 2011.

[2] A. Gandhi, M. Harchol-Balter, R. Raghunathan, and M. A. Kozuch, "AutoScale: Dynamic, robust capacity management for multi-tier data centers," *ACM Transactions on Computer Systems*, vol. 30, no. 4, pp. 14:1–14:26, 2012.

[3] L. Deboosere, B. Vankeirsbilck, P. Simoens, F. Turck, B. Dhoedt, and P. Demeester, "Efficient resource management for virtual desktop cloud computing," *The Journal of Supercomputing*, vol. 62, no. 2, pp. 741–767, 2012.

[4] 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.

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

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

TABLE VI
POWER CONSUMPTION AT DIFFERENT LOAD LEVELS IN WATTS

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

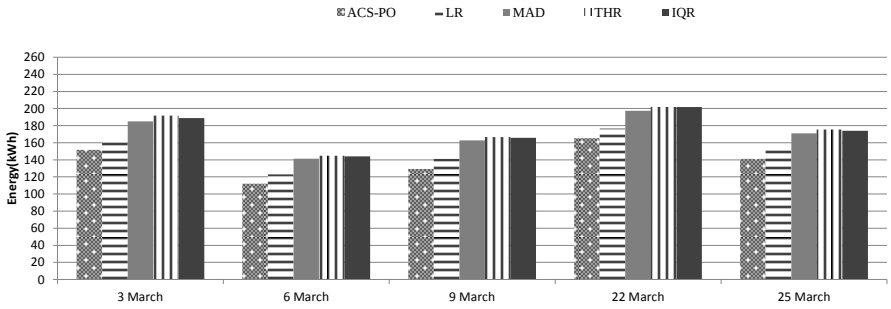

Fig. 3. Energy consumption by ACS-PO and benchmark methods based on the real workload from March 2011

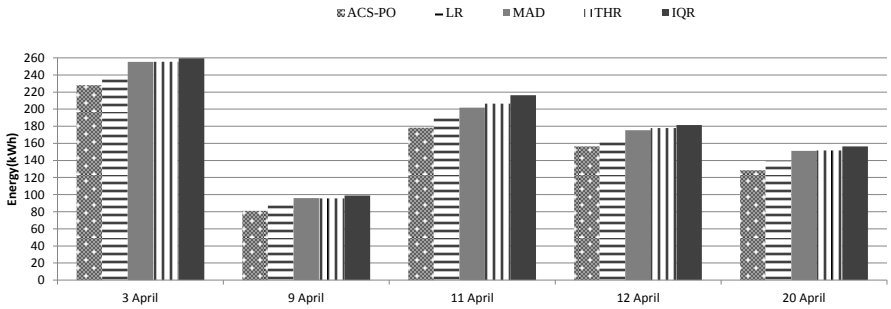

Fig. 4. Energy consumption by ACS-PO and benchmark methods based on the real workload from April 2011

[7] W. Vogels, "Beyond server consolidation," *ACM Queue*, vol. 6, no. 1, pp. 20–26, 2008.

[8] 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*, December 2012, pp. 26–33.

[9] A. Murtazaev and S. Oh, "Sercon: Server consolidation algorithm using live migration of virtual machines for green computing," *IETE Technical Review*, vol. 28, no. 3, pp. 212–231, 2011.

[10] M. Marzolla, O. Babaoglu, and F. Panzieri, "Server consolidation in clouds through gossiping," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, 2011, pp. 1–6.

[11] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Grid Computing and eScience, Future Generation Computer Systems (FGCS)*, vol. 28, pp. 755–768, 2012.

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

[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 *6th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, 2013.

[15] 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.

[16] Y. Ajiro and A. Tanaka, "Improving packing algorithms for server consolidation," in *Proceedings of the International Conference for the Computer Measurement Group (CMG)*, 2007, pp. 399–407.

[17] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *INFOCOM, 2011 Proceedings IEEE*, 2011, pp. 71–75.

[18] M. Harman, K. Lakhotia, J. Singer, D. R. White, and S. Yoo, "Cloud engineering is search based software engineering too," *Journal of Systems and Software*, vol. 86, no. 9, pp. 2225–2241, 2013.

[19] 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.

[20] M. Dorigo and L. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 53–66, 1997.

[21] P.-Y. Yin and J.-Y. Wang, "Ant colony optimization for the nonlinear resource allocation problem," *Applied Mathematics and Computation*, vol. 174, no. 2, pp. 1438–1453, 2006.

[22] A. Ashraf and I. Porres, "Using ant colony system to consolidate multiple web applications in a cloud environment," in *22nd Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, 2014, pp. 482–489.

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

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