This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2919555, IEEE Transactions on Services Computing

MAGNETIC: Multi-Agent Machine Learning-Based Approach for Energy Efficient Dynamic Consolidation in Data Centers

Kawsar Haghshenas, Ali Pahlevan, *Student Member, IEEE*, Marina Zapater, *Member, IEEE,* Siamak Mohammadi, *senior Member, IEEE,* and David Atienza, *Fellow, IEEE*

Abstract—Improving the energy efficiency of data centers while guaranteeing Quality of Service (QoS), together with detecting performance variability of servers caused by either hardware or software failures, are two of the major challenges for efficient resource management of large-scale cloud infrastructures. Previous works in the area of dynamic Virtual Machine (VM) consolidation are mostly focused on addressing the energy challenge, but fall short in proposing comprehensive, scalable, and low-overhead approaches that jointly tackle energy efficiency and performance variability. Moreover, they usually assume over-simplistic power models, and fail to accurately consider all the delay and power costs associated with VM migration and host power mode transition. These assumptions are no longer valid in modern servers executing heterogeneous workloads and lead to unrealistic or inefficient results. In this paper, we propose a centralized-distributed low-overhead failure-aware dynamic VM consolidation strategy to minimize energy consumption in large-scale data centers. Our approach selects the most adequate power mode and frequency of each host during runtime using a distributed multi-agent Machine Learning (ML) based strategy, and migrates the VMs accordingly using a centralized heuristic. Our Multi-AGent machine learNing-based approach for Energy efficienT dynamIc Consolidation (MAGNETIC) is implemented in a modified version of the CloudSim simulator, and considers the energy and delay overheads associated with host power mode transition and VM migration, and is evaluated using power traces collected from various workloads running in real servers and resource utilization logs from cloud data center infrastructures. Results show how our strategy reduces data center energy consumption by up to 15% compared to other works in the state-of-the-art (SoA), guaranteeing the same QoS and reducing the number of VM migrations and host power mode transitions by up to 86% and 90%, respectively. Moreover, it shows better scalability than all other approaches, taking less than 0.7% time overhead to execute for a data center with 1500 VMs. Finally, our solution is capable of detecting host performance variability due to failures, automatically migrating VMs from failing hosts and draining them from workload.

Index Terms—Host Power Mode, Machine Learning, Migration Cost, Power Mode Transition Cost, VM Consolidation, VM Migration, Energy Efficiency, Cloud Data Centers.

1 Introduction

High energy consumption and performance variability problems are two major challenges in modern cloud data centers, as they greatly affect operational expenses, total cost of ownership and revenue [1]. Global data center electricity usage accounted for 1.1-1.5% of total electricity use in 2010 [2] and increases at yearly rate of 2.1% [3]. As most data centers rely on fossil fuels as their main energy source, huge energy consumption also results in high carbon emissions and environmental concerns. However, according to a recent report by Shehabi et al. in 2016 [4], a potential of 45% reduction in electricity demand of data centers can be achieved compared to current trends, by improving their energy efficiency. Moreover, for the significantly large data centers, hardware failures and software anomalies are more frequent, leading to application performance variability and, eventually, Quality of Service (QoS) degradation [5].

- *K. Haghshenas and S. Mohammadi are with the School of Electrical and Computer Engineering, University of Tehran, Tehran 17469-37181, Iran E-mail: {khaghshenas, smohamadi}@ut.ac.ir*
- *K. Haghshenas, A. Pahlevan, M. Zapater, and D. Atienza are with the Embedded Systems Laboratory (ESL), EPFL, Switzerland E-mail: {kosar.haghshenas, ali.pahlevan, marina.zapater, david.atienza}@epfl.ch*
- *Corresponding author: Siamak Mohammadi.*

Manuscript received April 19, 2005; revised August 26, 2015.

To improve the energy and resource efficiency of virtualized data centers, the most common approach is exploiting Virtual Machine (VM) consolidation, a technique that tries to pack as many VMs as possible on one physical host. Furthermore, by switching idle hosts to low power modes, live VM migration allows to consolidate VMs dynamically to have a better VM placement and increase energy savings. In this sense, the low power modes of recent servers, more efficient in terms of energy and transition overheads [6], allow further savings when compared to traditional switch-off techniques. However, given the variable nature of VMs loads, and energy and delay overheads of both migration and Power Mode Transitions (PMTs), dynamic consolidation may degrade QoS (and even increase overall energy consumption) if not effectively applied [7].

Most previous works in this context are migration overhead oblivious [8], [9], [10], whereas just few of them estimate these costs and take them into account [11], [12], [13]. Nevertheless, in such cases they assume that hosts would be switched-off when they have no load [10], [14]. In this regard, some research proposes heuristics to appropriately set the hosts' power modes [15], [16], keeping idle hosts in intermediate sleep modes to wake them up quickly in the presence of new loads. Meanwhile E-eco [15] is the closest to the scope of our work, as it uses dynamic VM consolidation together with host power management. As shown in the results of this paper, our solution outperforms E-eco, increasing

energy savings by 15% while maintaining the same QoS, and is able to scale to larger scenarios. Moreover, as opposed to all other solutions in the state-of-the-art (SoA), our approach is also able to tackle performance variability detection and mitigation.

Application performance variability is one of the major challenges in data centers, and may originate from both internal and external sources, e.g., aged or failing hardware, thermal control, orphan processes or operating system issues [5]. Most of these anomalies lead to performance degradation during host operation [1] and, therefore, to QoS degradation. Detecting performance variability is usually accomplished by means of real-time continuous monitoring and analyzing system logs. However, as the size of data centers increases, the amount of metrics to be collected and analyzed is paramount. The manual analysis of these volumes of data therefore becomes impractical [17], requiring the development of techniques to automatically detect and isolate servers that exhibit a severe degree of performance variability.

Our work tackles the aforementioned challenges by proposing a comprehensive cost- and failure-aware VM migration and host power mode selection run-time management policy that manages VMs and hosts in a data center. Our main goal is to minimize energy consumption while meeting QoS requirements, keeping low the number of migrations and PMTs. At the same time, our approach is able to detect hosts under-performing or exhibiting any kind of performance variability due to hardware or software failures, and automatically drains the node from workload, migrating VMs to other hosts to avoid QoS degradation.

In particular, our centralized-distributed approach consists on two units: i) a distributed Machine Learning (ML)-based multi-agent power mode selection unit, and ii) a centralized heuristic migration unit. The power mode selection unit determines the optimum power mode of each host based on its CPU utilization and number of VMs running. Given that the problem to be solved is NP-hard [18], our solution is based on the Q-Learning (QL) technique, which belongs to the Reinforcement Learning (RL) category [19], [20]. RL is highly recommended for multi-agent systems as they do not need to model the environment, and allow agents to take actions while they learn [21]. In multi-agent systems, agents lack full information about their counterparts, and thus the multi-agent environment constantly changes as agents learn about each other and adapt their behaviors accordingly. After performing power mode selection phase, a centralized migration unit detects the over-utilized hosts and migrates VMs to allow driving the hosts to the power modes selected by the power mode selection unit. Finally, CPU frequency is adjusted using Dynamic Voltage and Frequency Scaling (DVFS) to further reduce energy. Throughout the paper, we refer to our Multi-AGent machine learNing-based approach for Energy efficienT dynamIc Consolidation as MAGNETIC.

In summary, the main contributions of our work are as follows:

- We propose a scalable centralized-distributed ML-based host power mode selection and VM migration policy to dynamically consolidate VMs in hosts, reducing energy by up to 15% when compared to existing SoA approaches, with only minimal QoS degradation ($< 1\%$ difference) (see Section 6.4). MAGNETIC considers the overhead of VM migration and host PMT and reduces them by up to 86% and 90% respectively when compared to SoA.
- Thanks to the distributed nature of the power mode selection unit, that utilizes a multi-agent QL algorithm, our solution outperforms all other approaches in the SoA in terms of scalability. Furthermore, MAGNETIC's run time overhead is below 0.7% and 2.7%, for data centers with 1500 and 3800 VMs, respectively.
- Additionally, MAGNETIC is able to detect performance variability due to hardware or software anomalies exhibited by hosts. In particular, we show how a power mode selection agent is able to detect performance variation, and migrate all VMs from a degraded host. As an example, power mode selection agent switches a host performing at 50% of its maximum capacity, into the low power mode, in less than 8 hours (i.e., 100 iterations) since the anomaly started (see Section 6.7). Moreover, during this period, the combined action of power mode selection unit and migration unit try to keep the QoS by migrating the VMs of the degraded host.
- Our solution is evaluated using power models validated against real traces from enterprise servers, and account for both migration and PMT overheads. The models and proposed techniques, as well as our baselines for comparison have been incorporated into the CloudSim simulator [22].

The remainder of this paper is organized as follows. Section 2 summarizes related work. Then Section 3 introduces the power model that we have used. Section 4 formulates the problem formulation, whereas Section 5 describes our proposed MAGNETIC approach. In Section 6, the experimental setup is described and results are provided. Finally, a summary of our conclusions is drawn in Section 7.

2 Related Work

2.1 Energy-aware Dynamic VM Consolidation

Previous approaches on energy-aware dynamic VM consolidation can be categorized based on: i) their capabilities for considering power and delay overheads associated with VM migration and PMT, and ii) their control policy level (centralized, distributed).

Most of the previously published papers in this regard have proposed centralized approaches to decide VM migrations and hosts' power modes [8], [9], [10], [11], [12], [13], [15].

Abdullah et al. [8] have proposed a method that is composed of Fast Best-Fit Decreasing (FBFD) and Dynamic Utilization Rate (DUR) algorithms to consolidate VMs dynamically. Also, the underutilized host selection in [8] is the same as Power Aware Best Fit Decreasing (PABFD) algorithm proposed by Beloglazov et al. [10], however, two different strategies are applied to detect over-utilized hosts in [8]. Beloglazov et al. [10] have introduced several dynamic consolidation algorithms which work along with PABFD algorithm. The main differences between these algorithms are their overloaded host detection and VM selection methods. A similar approach has been proposed by Kansal et al. [9] which applies a meta-heuristic approach to detect overloaded hosts and forces the maximally loaded VMs to migrate to the least loaded active hosts. These approaches fall short in considering migration overheads in their control policy of dynamic consolidation that can lead to higher migration counts.

Nguyen et al. [11] have proposed a VM consolidation algorithm with multiple usage predictions (VMCUO-M) to predict resource utilization, and characterize overloaded and underloaded hosts. Sercon [13] is another migration cost-aware algorithm, which minimizes the total number of active hosts together with the number of migrations, saving energy due to the lower number of migrations, but being still unaware of migration overheads. Verma et al. [12] proposed pMapper algorithm, which improves energy

efficiency by minimizing migration costs. All the above mentioned algorithms are PMTs overhead oblivious methods as they do not account for the power-delay costs associated with PMTs.

In contrast to all these works, Rossi et al. [15] propose a performance-aware energy efficient hosts management approach that switches some idle hosts into intermediate sleep power modes to improve performance. The authors also investigate the efficiency of various sleep power modes. However they disregard the impact of VM migration overhead.

Although most of the approaches in data center resource management are fully centralized, some papers have proposed distributed and semi-centralized approaches which lead to better scalability and shorter runtime. Feller et al. [23] have proposed a fully decentralized dynamic consolidation technique that is applicable to unstructured peer-to-peer networks. Their proposed approach is built on top of the Cyclon protocol [24] which allows to periodically construct time-varying randomized P2P overlays. Contrary to our work, this algorithm does not consider the overheads of migrations and PMTs and has been designed to increase *Turned-off* hosts with a lower number of migrations. Then, Wu et al. [14] propose a semi-centralized technique based on an Improved Grouping Genetic Algorithm (IGGA) which switches off idle hosts to save energy, and takes into account migration costs. The main bottleneck of IGGA is its large runtime overhead, especially for large-scale data center scenarios.

Finally, a lot of studies exist trying to improve energy efficiency of a data center. To the best of our knowledge, our proposed QL-based approach is the first work that manages both hosts and VMs in a semi distributed manner, taking into account the power and delay overheads associated with both migrations and PMTs.

2.2 Performance Variability Awareness

Application performance variation can be caused by software anomalies or hardware failures that can directly impact QoS.

Many works detect anomalies through log file analysis [25], [26]. Diagnosing anomalies and failures through monitoring infrastructures and analyzing log files is a difficult task due to the need to collect large amounts of noisy and high-dimensional data at runtime. Recent studies propose anomaly diagnosis methods via monitoring tools with different sampling time period (e.g., every one minute or greater sampling period). In this way, some works detect anomalies based on the changes on host resources utilization. For this purpose, they manually select the samples of power, file system as well as memory demands congestion [25], [26]. Nevertheless, these works do not take into account an automated method to efficiently tackle these problems.

Other works exist that propose statistical techniques and machine learning-based methods to detect some specific anomalies such as network congestion, operating system performance issue, temperature-related issues, or hard disk and memory access errors with high precision [27], [28], [29]. Xu et al. [27] present a framework to detect anomalies through statistical techniques and switch to backup VMs or servers in the presence of failure. resource allocation method using backup VMs and links. Ibidunmoye et al. [29] investigate using ML algorithms to predict QoS and detect performance degradation. However, these methods neither consider the availability of computing nodes nor react to anomalies for energy efficiency with respect to the power and delay overheads associated with both migrations and PMTs.

In contrast, in this paper we exploit VMs resource usage and their performance characteristics to diagnose anomalies instead of using log files. Our proposed approach detects performance degradation and migrates the degraded host's VMs to switch it into the low power modes.

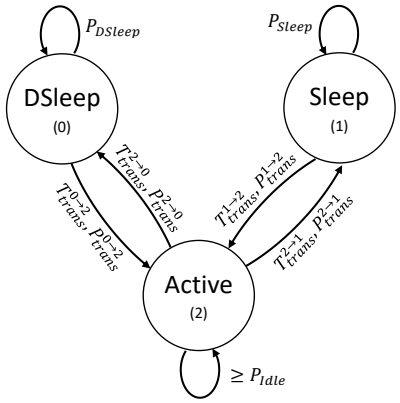

Fig. 1: Power modes transition periods and power consumption

3 Power Model of the Data Center

This section introduces the power models used in this paper to estimate data center energy consumption. In particular, we model overall data center power as the sum of the power consumption of all hosts (i.e., physical servers). Moreover, we account for the overhead of migrations by modeling the energy cost per VM, as described in the next subsections.

For the sake of clarity, Table 1 summarizes the main parameters and notations used throughout this paper.

3.1 Host Power Model

The power consumption of a host depends on both its specific hardware configuration and the utilization of its various processing components, including CPU, memory, hard disk, I/O, and network. Most power models estimate host power using a linear or square regression of CPU utilization. This estimation usually disregards memory power, and is only valid for one specific application, as shown in previous works [30], [31]. However, these models have been used by most of the works in the dynamic consolidation area, regardless of the application and workload traces, leading to over-simplified and unrealistic results, as we show in Section 6.

Hosts running modern operating systems can benefit from different power modes, which generally include: *Turned-off*, *Active*, *DeepSleep* (*DSleep*), and *Sleep* (see Fig. 1). A *Turned-off* host consumes zero power. Nevertheless, hosts are rarely *Turned-off* in today's data centers due to the very large time delay and energy loss of booting the server and resuming to *Active* mode. Therefore, we have not considered *Turned-off* as an available power mode in this paper (and therefore is not shown in Fig. 1). When a server is *Active* but not running any workload, we consider it is idle. Idle hosts are put into sleep modes (*DSleep* or *Sleep*) in order to save power. Sleeping hosts consume a constant power that is significantly lower than idle power. There are associated time delays and energy losses when a server is suspended to or resumed from these sleep modes. These are smaller when resuming from sleep to active, than when going from active to sleep. During the transition, the power consumption of the host for both suspending and resuming periods is similar and close to the maximum power of the host (P_{max}), but has a different delay overheads [16], [32]. Therefore, the energy consumption is $E_{trans} = P_{trans} \cdot T_{trans}$, where $P_{trans} \approx P_{max}$ and T_{trans} represent the power and delay overheads

TABLE 1: Overview of the used notation

General Parameters and Variables			
M	Number of VMs	$C^i_{Req^t}$	Requested computational resources of host i during time slot t
N	Number of hosts	$C^j_{Req^t}$	Requested computational resources of VM j during time slot t
P_i	Power of the physical host i	$C^i_{Alloc^t}$	Allocated computational resources of host i during time slot t
P_{Idle}	Physical host's idle power	$C^j_{Alloc^t}$	Allocated computational resources to VM j during time slot t
P_{Sleep}	Physical host's power in Sleep mode	mem^i_{max}	Maximum memory of host i
P_{DSleep}	Physical host's power in DeepSleep (DSleep) mode	C^i_{max}	Maximum computational resources of host i
$P_{dynamic^i_k}$	Dynamic power of the physical host i when running application k	$mem^j_{Alloc^t}$	Allocated memory to VM j during time slot t
P_{trans}	Physical host's power during the transition mode	$f_{i(t)}$	CPU frequency of host i during time slot t
T_{trans}	The latency of the physical host's transition mode	E^j_{mig}	Energy overhead of migrating VM j
E_{trans}	Physical host's energy consumption for a transition	V^j_{mig}	Network traffic of migrating VM j
$E_{DC(t_1,t_2)}$	Energy consumption of the data center's IT equipment during (t_1,t_2)	S_V	The set of all VMs of the data center
$SLA_{v(t_1,t_2)}$	SLA violation of the data center during (t_1,t_2)	S_H	The set of all hosts of the data center
E^i_t	Energy consumption of host i during time slot t		
Indices			
t	Index for time	j	Index for VM
i	Index for host	k	Index for application

of the transition, respectively, as shown in Fig. 1. *DSleep* is a sleep mode with lower power consumption compared to *Sleep* mode, whereas the delay and energy overheads when a server is suspended to or resumed from *DSleep* is higher than *Sleep* (see Section 6.1 for the specific values in our experimental setup).

Then, as power consumption of active hosts changes dynamically using DVFS and setting CPU resource limits with respect to the characteristics of the running workload, we use the power measurements presented in [33] to estimate the active power consumption of the considered host for different applications of the PARSEC benchmark suite [34]. For each application, the power consumption is measured as a function of CPU resources, i.e., the aggregated maximum computing capacity of the cores of a processor. For most PARSEC applications, there is a linear relationship between power and CPU resource limits, with the slope of the curve varying for each application. For most applications, the maximum power consumption happens at the maximum computing capability. However, for some of them that experience resource contention, after reaching a certain amount of CPU resources, power consumption remains constant [33].

If we consider that each VM runs one PARSEC application, static and dynamic consolidation algorithms lead to packing more than one application (i.e., VM) on one host. By running a variety of applications on a server, power consumption estimation gets even harder. As extending the power model to include all application mixes is unfeasible, we have estimated total server power consumption by adding the host's dynamic power consumption due to each application running on it. Dynamic power refers to the power usage of a host under a given load minus the host's idle power. Therefore, total power consumption of host i is calculated as follows:

$$P_i = P^i_{Idle} + \sum_{k=1}^{M_i} P^i_{dynamic}(U_k) \quad (1)$$

where P_{Idle} stands for host's idle power consumption. $P^i_{dynamic}(U_k)$ accounts for dynamic power of host i when running application k, and is proportional to the CPU resource utilization when running application k, (U_k). As we suppose that each VM runs one application, M_i stands for the number of VMs running on host i.

To compute the host's dynamic power for each application in PARSEC benchmark, the total CPU power has first been measured on the host for the different utilization rates (from 0 to 100%, discretized in 11 uniform levels). After removing the idle power from the measured total power for each utilization, dynamic power is computed. For values between two consecutive utilization levels, we use a linear interpolation to compute its dynamic power consumption. For PARSEC applications running on the server, described in Section 6.1, this linear model results in an error below 5% when compared to the real data [33].

Besides these power modes, DVFS is applied to active hosts to assign minimum voltage and frequency to each processor considering their requested computational resources.

3.2 VM Migration Energy Model

To compute the energy overhead of migration, we use a linear model presented in [35]. This model takes into account the network, as well as the energy consumption of the source and destination hosts. In a homogeneous network environment, the energy consumption for data transmission and reception are roughly the same when the data transmitted and received on source and destination hosts are equal. Therefore, the VM migration energy overhead is mainly determined by data transmission and reception over the networks. Hence, based on the experiments shown in [35], the energy consumption of migrating VM j consists of a linear component that increases with the traffic of VM migration over the network plus an offset which depends on the source and destination hosts, as follows:

$$E^j_{mig} = a \cdot V^j_{mig} + b \quad (2)$$

where network traffic V^j_{mig} is measured in Megabytes and energy E^j_{mig} is measured in Joules. Parameters a and b are the network and host configuration dependent constants, and assume that migration overhead is proportional to the amount of data volume and not related to the processing on the hosts. This linear model is proved to exhibit a maximum error of less than 10% [35]. In a live migration scenario, all the memory pages of the VM to be migrated are accessed and transferred through the network between the source and the destination hosts.

4 Problem Formulation of Host and VM Management

The problem of allocating VMs to physical hosts is referred to as VM allocation. Besides initial VM allocation, live VM migration is used to dynamically improve VMs placement. Over time, this procedure allows cloud providers to pack VMs into fewer hosts dynamically, and set more hosts into low power modes.

Furthermore, utilizing various sleep modes with different power consumption and inter-mode overheads, improves data center resource efficiency. Considering dynamic consolidation accompanied with host power mode selection in this paper, provides more opportunities for improving resource efficiency. In this way, the control knobs are VM migration and host PMT. Both operations are associated with power and delay overheads that should be considered in the optimization.

The first objective of our approach is increasing energy savings, which leads to packing more VMs on one host. As this procedure may affect QoS, minimizing Service Level Agreement (SLA) violation is considered as the second objective. We allocate host CPU capacity (C^j_{Alloc}, measured in million instructions per second, i.e., MIPS) to VMs, up to their current requested MIPS. Therefore, two conflicting objectives here are minimizing energy consumption and minimizing the difference between requested (C^j_{Req}) and allocated (C^j_{Alloc}) MIPS, which represents the SLA violation. Moreover, once a workload has been assigned a certain capacity, if the real MIPS measured differ from the allocated MIPS, we consider that the host is experiencing performance variability due to a hardware or software anomaly (i.e., the host is failing and workload should be migrated from it).

We formulate our optimization problem in equations (4)-(7). Let S_H and S_V represent the set of hosts and VMs of the data center, respectively, with N and M denoting their number of members during the time period (t_1, t_2) correspondingly. At each time t, each VM is running on a host that is described by $a_{ij}(t)$:

$$a_{ij}(t) = \begin{cases} 1 & \text{if } VM\,j \text{ is assigned to host } i,\ \forall j \in S_V\ \&\ \forall i \in S_H \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Based on the above explanations, we can formulate the optimization problem for the time period (t_1, t_2) as follows:

$$Min. \quad \sum_{t=t_1}^{t_2} E_{DC(t_1,t_2)} + SLA_{v(t_1,t_2)} \tag{4}$$

Subject to

$$\sum_{i=1}^{N} a_{ij}(t) = 1\ \forall j \in S_V \tag{5}$$

$$\sum_{j=1}^{M} C^j_{Alloc t} \cdot a_{ij}(t) < C^i_{max}\ \forall i \in S_H \tag{6}$$

$$\sum_{j=1}^{M} mem^j_{Alloc t} \cdot a_{ij}(t) < mem^i_{max}\ \forall i \in S_H \tag{7}$$

The first constraint (5) ensures that each VM is running only on one host, and the set of constraints (6) and (7) guarantee that the allocated CPU and memory resources from a host to its VMs is not higher than its capacity. C^i_{max} and mem^i_{max} represent maximum CPU and memory capacity of host i. The first and second terms of the objective function represent the total energy consumption of the IT equipment ($E_{DC(t_1,t_2)}$) and average SLA violation ($SLA_{v(t_1,t_2)}$) of the data center for the time period (t_1, t_2). $E_{DC(t_1,t_2)}$ can be defined as:

$$E_{DC(t_1,t_2)} = \sum_{j=1}^{M} E^j_{mig} \cdot \lambda^j_{mig} + \sum_{i=1}^{N} [P^i_{trans} \cdot T^i_{trans} \cdot \lambda^i_{trans} + P_i \cdot (t_2 - t_1 - T^i_{trans} \cdot \lambda^i_{trans})] \tag{8}$$

where the first term includes the energy consumption due to all migrations performed in the previous time slot. The second term represents the energy consumption during power mode transition phase, and the third represents the energy consumption of the host after setting the new power mode. Variables P^i_{trans} and T^i_{trans} represent power and delay overhead of power mode transition of host i. The binary decision variable λ^j_{mig} shows whether VM j has been migrated in the previous time slot or not. Also, the binary decision variable λ^i_{trans} indicates whether the power mode of host i has been changed in the previous time slot or not.

The SLA violation is defined based on the *'difference between requested and allocated MIPS'* of the VMs. Therefore, the second objective of the resource management optimization problem is represented as follows:

$$SLA_{v(t_1,t_2)} = \sum_{j=1}^{M} (C^j_{Req} - C^j_{Alloc}) \cdot (t_2 - t_1) \tag{9}$$

where C^j_{Req} refers to requested MIPS of VM j and C^j_{Alloc} stands for its allocated MIPS during time period (t_1, t_2).

The nature of the VM allocation problem along with host power mode selection is NP-hard. Therefore, potentially suboptimal solutions are required to obtain reasonable results in as short as possible runtime. In this paper, a machine learning based approach is proposed, where further details are provided in the next section.

5 MAGNETIC: Proposed Multi-Agent Machine Learning-Based Approach for Energy Efficient Dynamic Consolidation

Dynamic consolidation algorithms are designed to optimally migrate *'some VMs'* to *'destination hosts'* in order to improve energy efficiency. This paper proposes the MAGNETIC approach that, besides the dynamic consolidation, decides the *'appropriate power mode for all hosts'* in constant time intervals and also drains under performing hosts from workload, assuming that they may be experiencing hardware or software failure. In this way, our proposed approach manages VMs and hosts of the data center simultaneously at constant time intervals. Indeed, the number and the resources of the VMs do not change between time slots and VMs are only created or destroyed at the beginning of time slots. This not only leads to a better placement and configuration in terms of energy consumption, but also prevents unnecessary VM migrations and PMTs, as well as reliability issues. Furthermore, our proposed approach considers power and delay overheads related to VM migrations and host PMTs, which results in minimizing number of migrations and PMTs.

This centralized-distributed management approach is comprised of distributed per-host power mode selection agents, together with a central unit that decides VM migrations. First, the power mode selection unit chooses the most appropriate power mode for each of the hosts through a multi-agent QL-based algorithm, improving the trade-off between energy consumption and

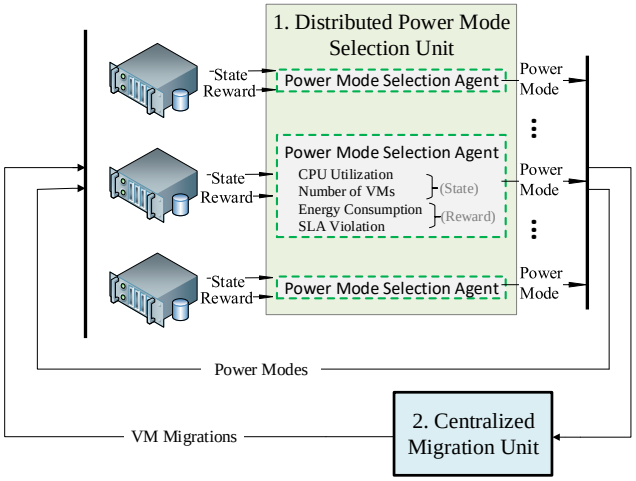

Fig. 2: Structure diagram of the proposed approach

SLA violation. Then, the migration unit performs the necessary migrations to satisfy these power modes. The migration unit also detects over-utilized hosts and forces some of their VMs to migrate to resolve their overutilization. Fig. 2 shows a diagram of the proposed approach, containing the above mentioned operations, which are applied in constant time intervals. The concepts of agent, state, and reward in this figure are explained in the following subsections. Next, we describe in detail the two units of the proposed approach and investigate their computational complexity.

5.1 Host Power Mode Selection Unit

Power mode selection unit consists of the ML-based distributed learning agents, one per host, each of which decides the optimal power mode of one host. Our proposed multi-agent power mode selection unit, besides eliminating centralized management strategy, which is practically impossible for large scale data centers, benefits from the large amount of workloads, and therefore different states, available in data centers, to select the optimal actions.

As there are no labeled training examples that power mode selection agents could use to learn, we use QL, which belongs to the semi-supervised learning algorithms category known as RL [19]. RL algorithms have the ability to find the optimal policy in a trial and error manner by observing the environment rewards. General RL frameworks consist of three main components: state, action, and reward. An agent selects an optimal action considering its state and receives a reward from the environment related to its selected action. The received reward shows the success/failure rate of the agent's decision. RL techniques are highly recommended for multi-agent systems because they neither need environment nor neighbors models. They allow an agent to take actions while it learns based on just its own state [21], [36], [37]. In a multi-agent RL approach, agents do not have full information about their counterparts and environment, so the multi-agent environment constantly changes as agents learn about each other and adapt their behaviors accordingly. In a multi-agent system, it is possible to apply QL in a straightforward fashion to each agent [21].

5.1.1 Q-Learning Basic Concepts

In QL algorithm, a Q value is associated with each state-action pair (s,a) and each state-action pair (s,a) creates an entry in a Q table that is initially set to zero. The Q value of each state-action pair (s,a) is updated every time the agent observes a state s and takes an action a as the optimal action. In this way, the optimal action $a*$ is selected by:

$$a* = argmax_{a_i \in A} Q(s,a_i) \tag{10}$$

where A represents the set of actions that are available at state s. As the process continues, each agent improves its current policy by updating the Q values stored in the Q table. If the initial Q function $Q_0(s,a)$ at time slot 0 is defined as:

$$Q_0(s,a) = \Psi(s,a) \tag{11}$$

for $t = 0,1,2,...$, the Q value of a (s,a) pair is updated by:

$$\begin{aligned} Q_{t+1}(s,a) \leftarrow [&(1-\alpha_t(s,a)) \cdot Q_t(s,a) + \\ &\alpha_t(s,a) \cdot (r_{t+1}(s,a) + \gamma \cdot argmax_{a_i \in A} Q(s',a_i))] \end{aligned} \tag{12}$$

where γ represents the discount factor and $\alpha_t(s,a)$ is the learning rate of the agent. The discount factor γ is in the range of [0, 1] and is used to determine the effect of near-future information. If γ is closer to 1, the weight of future reinforcements will be higher.

Learning Rate Discussion: learning rate $\alpha_t(s,a)$ determines how newly acquired information overrides old information. Setting $\alpha_t(s,a) = 1$ makes the agent focus on the immediate reward $r_{t+1}(s,a)$ and the estimated total future rewards $argmax_{a_i \in A} Q(s',a_i)$ [19]. Therefore, if we want agents to learn as they act, α_t needs to be a constant value, thus the agent learns from newly acquired information regardless of the number of visits. Nevertheless, if we prefer that, after a certain time, the agent stops learning, we can define the learning rate as the inverse function of the number of visits. More specifically, if we define $\alpha_t(s,a)$ as:

$$\alpha_t(s,a) = 1/(1 + Visits_t(s,a)) \tag{13}$$

where $Visits_t(s,a)$ indicates the number of preceding selections of action a in state s, it can be shown that when $\alpha_t(s,a) = 1/95 \approx 0.01$, the results from new visits hardly change the Q values that already have been learned [21]. As shown in the result section, setting the learning rate as in Equation (13) yields the best results in terms of energy efficiency. However, setting it to a constant value, enables our technique to detect host performance variability. As the defined reward function in this paper consists of allocated MIPS to the hosts' running VMs, the agents can observe the performance variation revealed in allocated MIPS, and if we use a constant value for $\alpha_t(s,a)$ parameter, agents can influence observed anomalies in their action selection polices.

We consider two phases to train the reinforcement model: (1) exploration and (2) exploration-exploitation. In the exploration phase, at the first time slot, each agent selects an action randomly from an action pool, which includes all the available power modes. Thereafter, the new Q-value corresponding to the selected action and the initial state is calculated. In subsequent time slots, when a state is observed again, the action is selected among a subset of actions that has not been selected yet. In the exploration-exploitation phase, the QL algorithm makes an optimal balance between exploration and exploitation. $\varepsilon - greedy$ is one of the widely used methods for exploration/exploitation balancing [38]. In this method, at each time slot, the agent selects a random action with the probability of $0 < \varepsilon < 1$ instead of selecting the learned optimal. This action selection policy is defined as follows:

$$\pi(s) = \begin{cases} argmax_{a_i \in A} Q(s,a_i), & \text{if } \xi < \varepsilon \\ random\ action\ from\ A, & \text{otherwise} \end{cases} \tag{14}$$

Algorithm 1 Host Power Mode Selection Unit Procedure

Start of the time period $(t, t+1)$
HostList: all hosts
for $i = 1 : N$ **do**
  S $\leftarrow$ Observe current state of host i.
  $\xi \leftarrow$ Generate a random number between 0 and 1.
  a* $\leftarrow$ Select optimal action using $\pi(s)$ (Equation 14).
end for
End of the time period (t,t+1)
for $i = 1 : N$ **do**
  r $\leftarrow$ Receive reward host i using (Equation 17).
  S' $\leftarrow$ Observe subsequent state of host i.
  Update the Q function using (Equation 12).
end for

where ξ is a uniform random number drawn at each time slot and can balance the trade-off between exploitation and exploration so that the algorithm converges to the optimal policy.

5.1.2 State and Action Definition

To apply QL for selecting host's power mode, we need to define the state space and the set of available actions. The state of an agent is defined by the state variables: CPU utilization (U_t) and number of VMs (N_{VMt}). Therefore, the state space S can be defined by:

$$S = \{s | s_{x,y} = (U_t, N_{VMt}), x = 0 : U_1 \quad and \quad y = 0 : U_2\} \tag{15}$$

The range of host's CPU utilization is [0,100]%. We map this utilization onto the discrete range of $[0, U_1]$ where $x = 0$ and $x = 10$ corresponds to the CPU utilization of 0% and 90-100%, respectively. The value of the variable y is determined by the number of VMs on the host. In this paper, we have set U_1 and U_2 to 10 and 1, respectively. Hence, CPU utilization discretization is done in steps of 10, and variable y is defined as follows:

$$y = \begin{cases} 0 & \text{if } N_{VMt} \leq 2 \\ 1 & \text{otherwise} \end{cases} \tag{16}$$

The variable y forces the agent to consider the number of running VMs on its corresponding host in its action selection policy. In this way, the agent distinguishes between the cases where the host has the same CPU utilization but the number of running VMs is different. As selecting an inappropriate action for a host with more running VMs causes higher SLA violation, we have heuristically selected a small enough value, 2 (in (16)), for the boundary between high number and low number of VMs.

At each state s, the available action set for the agent is $\{Active, Sleep, DSleep\}$ which represents the available power modes. Therefore at each time slot, the agent decides to drive the host to one of the power modes. Algorithm 1 reviews the described process of host power mode selection unit.

5.1.3 Reward Function Definition

The main objective of server consolidation is energy savings. However, maximizing energy savings usually degrades QoS. Therefore, QoS has to be considered as a second objective (or a constraint) in the optimization problem. We consider energy consumption and QoS two conflicting objectives and involve these objectives in the reward function of the learning process.

As mentioned before, SLA violation is defined based on the difference between requested and allocated CPU resources (see Equation 9) to its running VMs. Hence, SLA violation is zero if the MIPS allocated to the VMs of a host ($C^i_{Alloc^t}$) is equal to or higher than its requested MIPS ($C^i_{Req^t}$). To consider both energy consumption and SLA requirements objectives, this paper proposes the following reward function:

$$r^i_{t+1} = \begin{cases} (-1) \cdot N(E^i_t) & \text{if } C^i_{Req^t} = 0 \} \quad \text{Item 1} \\ c_1 \cdot N(E^i_t) - c_2 \cdot N(C^i_{Req^t}) & \text{if } C^i_{Req^t} > 0 \ \& \ C^i_{Alloc^t} = 0 \} \quad \text{Item 2} \\ c_1 \cdot N(E^i_t) + c_2 \cdot N(C^i_{Alloc^t}) & \text{otherwise} \} \quad \text{Item 3} \end{cases} \tag{17}$$

where E^i_t represents the total energy consumed by host i in the last time slot t ($N(E^i_t)$ is normalized E^i_t to $[0,1]$). This reward is forwarded to agent i at the beginning of time slot t+1.

The three items of the reward function cover different host's states. The first corresponds to the case where the host has no running VMs. The second is related to the case where the host has been switched to *Sleep* or *DSleep* while hosting some VMs. In this case, the agent gets a penalty proportional to the difference between requested and allocated MIPS. For a host driven to a sleep mode, higher $C^i_{Req^t}$ produces lower reward and consequently teaches the agent to not drive the host to a sleep mode when $C^i_{Req^t}$ is high. Finally, the third term helps the agent to drive the host to higher utilization to get a higher reward. Actually, the difference between the second and third items is on $C^i_{Alloc^t}$ that must be zero for the second item (host has been driven to a sleep mode) and higher than zero for the third item (host is in *Active* mode). As the agents desire to decrease energy and increase allocated MIPS, c_1 and c_2 in the reward function have to be in the ranges of [-1, 0] and [0, 1], respectively. These two weights are used to improve the trade-off between objectives. The energy part of the reward function, E^i_t, consists of three terms:

$$E^i_t = P^i_{trans_t} \cdot T^i_{trans_t} \cdot \lambda^i_{trans} + P^i_t \cdot (T_{timeslot} - T^i_{trans_t} \cdot \lambda^i_{trans}) + 1/2 \cdot E^i_{migrations} \tag{18}$$

The first represents the energy overhead of the host power mode transition. The second concerns the energy consumption of host i, after setting in the new power mode during time slot $(t, t+1)$. $T_{timeslot}$ stands for considered time interval. Finally, the third term aims to involve the energy overheads of incoming and outgoing migrations of host i ($E^i_{migrations}$). As for each VM migration both source and destination hosts are involved, we consider overheads are equally split on the rewards of these hosts.

5.2 Migration Unit

To improve VMs placement, the migration unit migrates VMs considering the selected power modes for the hosts taken from power mode selection unit. Moreover, the migration unit migrates some VMs from overloaded hosts in order to meet QoS requirements. A host is considered overloaded if its capacity is lower than the aggregated requested MIPS of its VMs (i.e., $C^i_{max} < C^i_{Alloc}$). Our proposed approach uses a static threshold technique to detect overloaded hosts [30].

Power mode selection unit may change a host's mode from *Active* to *Sleep*/*DSleep*. In this case, VMs on this host have to be migrated to other active hosts during the next time slot. The main difference between the proposed approach and previously presented dynamic consolidation algorithms is the underutilized host selection method. MAGNETIC selects some hosts to migrate

all of their VMs considering their selected power modes. More specifically, a host is considered underutilized if its power mode is changing from *Active* to *Sleep/DSleep* for the next time slot.

Thereafter, migration unit selects the destinations of VM migrations. Destination host selection is an online bin packing problem which can be solved using heuristics like Best Fit (BF), First Fit (FF), and Best Fit Decreasing (BFD) algorithms. Due to the low complexity and low computational overhead of BF, we use a modified BF algorithm to select the destination hosts of VM migrations for the next time slot. Although using more complicated consolidation methods, including load balancing, improves VMs placement on active hosts, it also increases the number of VM migrations and may interfere with the learning process of hosts' power modes selection agents.

5.2.1 Migration Algorithm Description

Algorithm 2 reviews the explained operations, which can be summarized in three main steps. First, overloaded hosts are detected using a static threshold (U_t) heuristic. Migration unit selects VMs to be migrated from overloaded hosts considering their selected power modes (Lines 1-11). If the selected power mode for an overloaded host is *Active*, its VMs are chosen to be migrated to reduce overutilization. On the contrary, if its selected power mode is a sleep mode (this may happen during the exploration phase of the training), all of its VMs are migrated. The point here is that, as the energy and delay overheads of migrating all VMs of an overloaded host is high, the reward of its associated agent is low and the probability of selecting this action in the future decreases. Second, the algorithm migrates all VMs from hosts whose power modes are changing from *Active* to *Sleep/DSleep* (Lines 12-28). Affected VMs are sorted in decreasing order (according to their C^i_{Req}) to be migrated to active hosts. To select the destinations of these migrations, the algorithm uses a modified BF method. To this purpose, the algorithm first selects hosts that are transitioning from *Sleep/DSleep* mode to *Active* mode and have no running or incoming VMs, and then selects hosts based on their available MIPS (C^i_{Avail}) by the BF method. The reason for this prioritization, is to migrate at least one VM to each of the hosts whose power modes are changing from *Sleep/DSleep* to *Active*, to provide a correct reward for their agents.

Despite the previous policy, there may still be some hosts that have just transitioned to *Active* mode and do not have any running or incoming VMs. Therefore, in the third step of the algorithm, migration unit migrates one VM to each of these hosts (lines 29-38). VMs are selected to be migrated to these hosts based on the available MIPS after being de-allocated from their current hosts.

After applying the selected power modes and VM migrations, the frequency of host i at time slot t is adjusted based on the allocated resources of its running VMs, $C^i_{Alloc^t}$, by:

$$f_{i(t)} \leq \frac{C^i_{Alloc^t}}{C^i_{max}} \cdot f^i_{max} \tag{19}$$

where C^i_{max} and f^i_{max} represent the maximum computational capacity and maximum CPU frequency of host i, respectively.

The computational complexity of our algorithm is $O(N_{Active} \cdot M_{mig})$ where N_{Active} and M_{mig} represent the number of migrating VMs and number of active hosts of the data center, respectively. This complexity is better than the popular complexity of most of the algorithms in this area which is $O(N.M)$, (N and M represent the number of hosts and VMs of the data center, respectively).

Algorithm 2 VM Migration Unit Procedure

1: $OverLoadedHosts \leftarrow$ Detect overloaded hosts using U_t
2: **for** all $OverLoadedHosts$ **do**
3: **if** host is *active* in next time slot **then**
4: **while** host is overloaded **do**
5: $SelectedVm \leftarrow$ VM with minimum C^j_{Req}
6: Add $SelectedVm$ to $MigratingVMs$
7: **end while**
8: **else**
9: Add all VMs from host to $MigratingVMs$
10: **end if**
11: **end for**
12: Add all VMs from hosts whose power mode is changing from *Active* to *Sleep/DSleep* into $MigratingVms$
13: Sort $MigratingVms$ in decreasing order
14: **for** all $MigratingVms$ **do**
15: **for** all *hosts* **do**
16: **if** host has no running or incomming VM **then**
17: $SelectedHost \leftarrow$ host
18: **end if**
19: **end for**
20: **if** $SelectedHost$ is $NULL$ **then**
21: $minC_{Avail} \leftarrow 0$
22: **for** all *hosts* **do**
23: **if** C^i_{Avail} after allocation of VM $\leq$ minC$_{Avail}$ **then**
24: $SelectedHost \leftarrow$ host
25: $minC_{Avail} \leftarrow C^i_{Avail}$
26: **end if**
27: **end for**
28: **end if**
29: **end for**
30: $EmptyHosts \leftarrow$ Detect hosts that have no running/incoming VM
31: **for** all $EmptyHosts$ **do**
32: $minC_{Avail} \leftarrow 0$
33: **for** all VMs **do**
34: **if** VM is not in migration **then**
35: **if** C^i_{Avail} of VM's host after deallocation $\leq minC_{Avail}$ **then**
36: $SelectedVM \leftarrow$ VM
37: $minC_{Avail} \leftarrow C^i_{Avail}$
38: **end if**
39: **end if**
40: **end for**
41: **end for**
42: Set best frequency for all active hosts

6 Experimental Setup and Results

This section compares our solution with SoA approaches in different perspectives. Section 6.1 describes experimental setup of the simulations. Section 6.2 represents the parameter tuning process of the learning agents. The proposed and compared approaches are evaluated in terms of energy consumption, average SLA violation, and runtime in Section 6.3. Also, we investigate the overheads and the scalability of these solutions in sections 6.2 and 6.3, respectively. Finally, Section 6.7 evaluates the capability of our proposed approach to detect performance degradation.

6.1 Experimental Setup

To evaluate the effectiveness of our solution, we use an enhanced version of the CloudSim simulator [22]. We extend CloudSim to support the power models and workloads introduced in Section 3 and implement our solution and the baseline for comparisons that are explained next (Section 6.3). A computer with a Core i7, 3.6 GHz CPU and 32GB RAM has been used to run the simulations.

We consider a data center with 1600 hosts and a shared storage infrastructure. The number of used hosts after dynamic consolidation is much lower than 1600, however with this setup we can use the same initial VM allocation for all considered benchmarks (with different number of VMs). To make a fair comparison, we assess energy savings with respect to other approaches in the SoA, and not against the initial number of the used hosts. Table 2 describes the characteristics of the simulated physical host, an AMD Magny Cours (Opteron 6172) with 12 cores on a single chip.

Solutions have been evaluated using different real workload traces provided as a part of the CoMon project [39]. In this project, CPU utilization has been obtained every 5 minutes from more than a thousand PlanetLab VMs hosted on more than 500 hosts. Therefore, 5 minutes is the time interval selected to apply dynamic consolidation, in our work. This time interval depends on the nature of the applications running on the data center. We have selected five days of traces collected during March and April 2011 of the CoMon project to cover different number of VMs and resource utilization patterns (which leads to different CPU utilization mean and standard deviation). The characteristics of the workload traces are shown in Table 3. As for all the PlanetLab traces the mean CPU utilization is low, we have simulated VMs with different number of cores to cover all the host's utilization levels from zero up to its capacity. The name of each workload trace represents the day and month of collecting data. VM utilization traces have been randomly mapped to the PARSEC benchmarks in order to emulate various VM types that run applications of different nature and therefore different in their CPU and memory requirements, as well as on their power consumption.

We have considered three different power modes in this paper; *Active*, *Sleep* and *DSleep* (see Section 3 and Fig. 1).

Equation 1 and the measurements of [33] are used to calculate active host's power consumption. For these measurements, each application runs on an isolated VM in a virtualized testbed using the VMware vSphere hypervisor. The power consumptions of *Sleep* (P_{Sleep}) and *DSleep* (P_{DSleep}) hosts are extracted as 25% and 5% of P_{max}, respectively [40]. Transition power and delay overhead between the power modes, as shown in Table 4, are obtained from the combination of two models presented in [40] and [32] using a linear regression. Finally, for the migration energy overhead, we derive the parameters $a = 0.461$ and $b = 18.127$, linearly interpolating between our considered host power values and the one in [35].

TABLE 2: Characteristics of the used host

Type	Value
Number of Cores	12
Memory (GB)	16GB
Peak Power (W)	165W
Idle Power (W)	66W
MIPS per Core	2000

TABLE 3: Characteristics of workload traces

Characteristics	03/06	04/12	04/11	04/09	04/03	03/22
Number of VMs	898	1054	1233	1358	1463	1516
CPU Util. Mean (%)	11.44	11.54	11.56	11.12	12.39	9.26
CPU Util. St.dev (%)	16.83	15.15	15.07	15.09	16.55	12.78

TABLE 4: Characteristics of the used host

	P_{trans} (W)	T_{trans} (s)
DSleep to *Active*	P_{max}	200
Active to *DSleep*	P_{max}	55
Sleep to *Active*	P_{max}	10
Active to *Sleep*	P_{max}	85

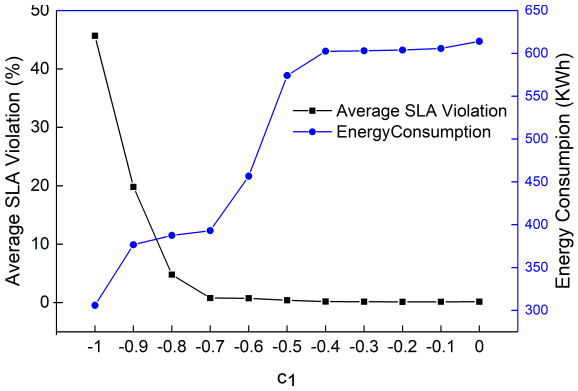

Fig. 3: Energy consumption and average SLA violation for different values of c_1

6.2 Parameter Tuning for Q-Learning

As explained in Section 5.1, parameters γ, α, ε, c_1 and c_2 have to be tuned based on the optimization objectives and the environment characteristics. The discount factor γ and ε have been chosen experimentally and have been set to 0.8 and 0.001, respectively. For all the experiments in this section, we set the α parameter as the inverse function of the number of visits (as in Equation (13)), except for Section 6.7 where we set it to a constant to investigate host performance degradation.

The parameters c_1 and c_2 in the reward function represent scalar weights of our two conflicting objectives in the learning process. Parameter c_1 has been swept from -1 to 0 (which results in sweeping c_2 from 0 to 1), to analyze the impact of these weights on the energy consumption and average SLA violations. As Fig. 3 shows, the energy consumption of the data center decreases by increasing the weight of energy (the absolute value of c_1) in the reward function. Also, increasing the weight of allocated MIPS and the penalty of violated requested MIPS (c_2), improves QoS.

The shape of the curves for energy and SLA violation when sweeping c_1 is not symmetrical. This is because Fig. 3 represents the energy consumption and average SLA violation of the whole data center, while c_1 and c_2 are the coefficients used in the reward function of each agent. Indeed, the effect of the used parameters (E_t^i, $C_{Req_t}^i$, and $C_{Alloc_t}^i$) in the reward function on the energy consumption and average SLA violation is not the same.

6.3 Compared SoA Approaches

In order to evaluate MAGNETIC, we have compared it with several existing approaches, ranging from non-power aware solutions to the SoA in the field, in terms of energy consumption, SLA violation, runtime, Migration and PMT count overheads, scalability, and the ability of performance degradation detection:

- Non-power aware solution (NPA): in this algorithm, VMs are running on hosts based on the initial VM allocation algorithm

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2919555, IEEE Transactions on Services Computing

and no dynamic consolidation approach is used to improve VMs placement. The only energy saving technique is DVFS.

- PABFD [10]: The power-aware best fit decreasing algorithm improves VMs placement by migrating VMs from underutilized hosts. PABFD optimizes VMs placement every 5 minutes by recognizing over-utilized and underutilized hosts. This algorithm detects over-utilized hosts using a static threshold and selects a VM to migrate from it with minimum CPU utilization and migrates VMs while its over-utilization is resolved. To improve VMs placement of underutilized hosts, PABFD looks for the host with the minimum CPU utilization, migrates all its VMs, and switches it to *Turned-off* power mode. For each migrating VM, PABFD selects a destination host with the minimum power increase due to the VM reallocation. Any underutilized active host which can host the VM can be a candidate destination. In this approach, a host is in *Active* mode if its CPU utilization is higher than zero, otherwise it will be in *Turned-off* mode.
- IGGA [14]: The dynamic consolidation algorithm proposed in [14] migrates VMs based on a genetic algorithm that tries to minimize power consumption and migration overheads. In this way, in each iteration, a VMs placement is chosen based on a defined score and using an improved genetic algorithm. The defined score is a linear function of power and data transmission overheads of migrating to the new VMs placement. Similar to PABFD, IGGA switches idle hosts to *Turned-off* mode, regardless of host PMT overheads. The main disadvantage of IGGA is its huge runtime.
- E-eco [15]: This approach manages VMs placement and hosts' power modes to decrease energy consumption and improve performance. E-eco is aware of host PMT overheads, and each host can be in *Running* (*Active*), *Sleep*, or *Turned-off* mode at each time slot. E-eco keeps some hosts in *Sleep* mode to turn them on quickly when needed. Different energy-saving techniques (dynamic consolidation or DVFS) are evaluated for different VM's sizes and placements and are used in E-eco to improve the trade-off between energy consumption and performance. E-eco uses several sub-routines to select power modes, energy-saving technique, and CPU frequency for each host. This algorithm is centralized and is not aware of migration cost, leading to high number of migrations.

For a fair comparison, when implementing these algorithms, we have considered *DSleep* instead of *Turned-off* (or *Switch off*).

6.4 Energy, QoS, and Performance Evaluation

This section compares the above mentioned algorithms in terms of energy consumption, average SLA violation and runtime using different benchmarks. We have applied dynamic consolidation for a 24-hour period of data center operation and reported the mentioned parameters for the last 23 hours, to limit the impact of initial VM allocation policy on the results. Normalized energy consumption of the data center for different algorithms to the NPA, running each of the 6 workload traces, are shown in Fig. 4. X-axis of this graph represents workload traces in increasing order in terms of the number of VMs. Fig. 4 shows that MAGNETIC achieves minimum energy consumption for all traces (up to 15% energy saving). The energy savings of our proposed approach increase with the number of VMs, thus showing better scalability. We discuss this important characteristic in detail in Section 6.6.

Then, Fig. 5 shows the average SLA violation of the data center due to the different algorithms. Although MAGNETIC

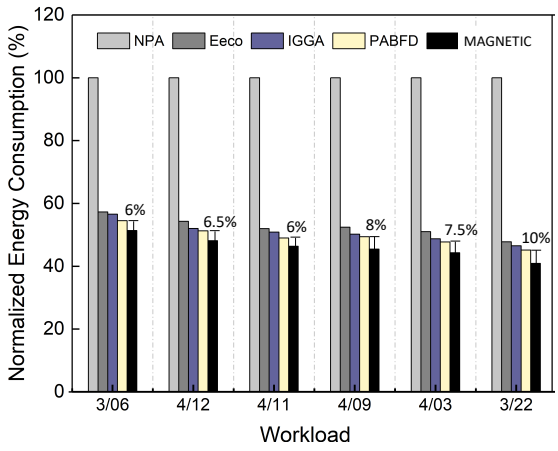

Fig. 4: Normalized Energy consumption to NPA for 23 hours operation of the modeled data center

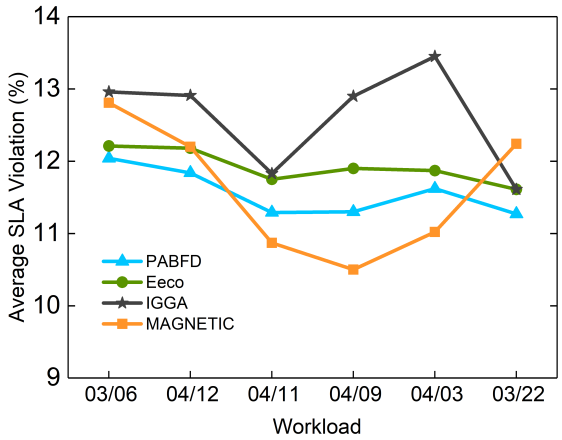

Fig. 5: Average SLA violation for 23 hours operation of the modeled data center

improves energy efficiency further than the compared algorithms, it exhibits similar or even less SLA violation. For each workload trace, the difference between MAGNETIC and the best algorithm in terms of average SLA violation, is less than 1%. The trade-off between energy and QoS in our algorithm is taken into consideration by experimentally adjusting the c_1 and c_2 variables of the reward function. Furthermore, selecting an appropriate training data set that matches the data center application can improve the results of the MAGNETIC for both the energy consumption and the SLA. Hence, better results could be derived in real implementations, with more effort on the learning part of the algorithm. As NPA does not use migrations to improve VMs placement, the number of migrations obtained by this algorithm and consequently its SLA violation are zero. Hence, the results of this algorithm are not shown in Fig. 5 and Fig. 6.

We have measured the runtime of different algorithms over a 23 hours data center operation, and depicted the average value for a single run in Fig. 7. Although the required runtime for our proposed approach is higher than NPA, PABFD, and E-eco, for less than 1 seconds, this runtime is below 2 seconds for all workloads, which is lower than the dynamic consolidation intervals (dynamic consolidation is applied with 5 minutes time intervals), representing an overload of less than 0.7%. The runtime of IGGA is much higher than that of other algorithms, making it

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2919555, IEEE Transactions on Services Computing

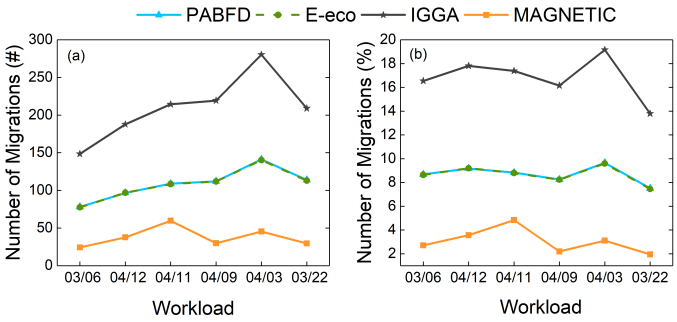

Fig. 6: Number of migrations in each time slot for a 23 hours operation of the modeled data center. (a) Average number of migrations per time slot (b) The percentage of average number of migrations to total number of VMs per time slot

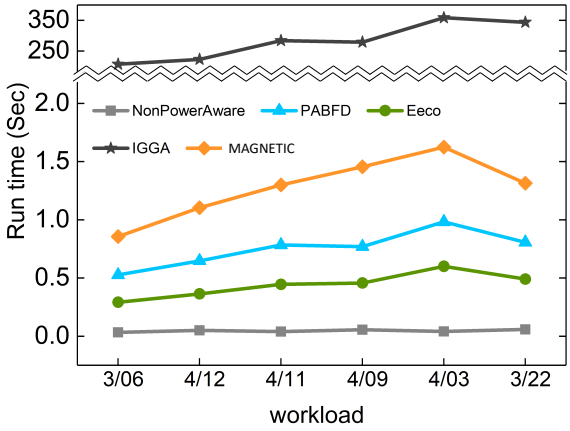

Fig. 7: Average runtime of each approach

an unfeasible approach, especially for large-scale data centers.

6.5 Migration and PMT count overheads analysis

Dynamic consolidation approaches utilize VM migrations and host PMTs to improve VMs placement and resource efficiency. These operations are associated with energy and delay overheads. Even if the management algorithm considers these overheads in its policy, algorithms with fewer migrations and PMTs are preferred in practice. This is due to the potential underestimation and inaccuracies when modeling overheads. In particular, it is the effect of network bandwidth overhead, which is not generally accounted for and could be considerable for large scale data centers. These two factors increase the chance of approaches with fewer number of migrations and PMTs to be used in real data centers. Therefore, this section evaluates MAGNETIC and compared algorithms in terms of number of migrations and number of PMTs.

Fig. 6 shows the average number of migrations in each time slot, during a 23 hours of data center operation, utilizing the proposed and compared approaches. MAGNETIC decreases the number of migrations by 45%-73%, 44%-73%, and 72%-86% when compared to PABFD, E-eco, and IGGA, respectively. Although for most of the dynamic consolidation algorithms higher migrations result in better placement, our approach achieves higher energy efficiency with much lower number of migrations.

Table 5 shows the number of PMTs caused by different algorithms over 23 hours of data center operation for the workload trace 03/06 with 898 VMs (the smallest workload trace). Our approach decreases the number of PMTs 52%-90%, when compared to other algorithms. This table also shows that our algorithm has used all available power modes while PABFD and IGGA (that always switch off idle hosts), only use *Active* and *DSleep*.

TABLE 5: Number of PMTs for different algorithms

Transitions	NPA	PABFD	E-eco	IGGA	MAGNETIC
All	270	393	652	1382	127
Active to DSleep	270	203	195	698	32
Active to Sleep	0	0	0	0	19
Sleep to DSleep	0	0	67	0	15
Sleep to Active	0	0	147	0	13
DSleep to Active	0	190	35	684	39
DSleep to Sleep	0	0	208	0	9

6.6 Scalability Evaluation

This section investigates the scalability of MAGNETIC when compared to others. The largest real traces in terms of the number of VMs in PlanetLab consist of 1463 VMs. Therefore, to investigate scalability, we have combined the available traces to produce larger ones, obtaining synthetic traces shown in Table 6.

As IGGA exhibits runtimes above the duration of the time slot and is unfeasible for large scale scenarios, we do not repeat the results for this algorithm. Table 7 represents the energy consumption and the average SLA violation of the data center using different algorithms. MAGNETIC decreases the energy consumption by around 58%, 10%, and 15% compared to NPA, PABFD, and E-eco, respectively. While having the same amount of SLA violation (less than 1% difference). Table 8 represents runtime and number of migrations of different algorithms. As mentioned before, NPA does not improve VMs placement. Hence, the number of its migrations is zero and its runtime for large scale data centers is as low as for small scale cases. The main achievement of the proposed algorithm for large scale data centers is its very low number of migrations while improving the energy efficiency. Our proposed algorithm decreases the number of migrations by 83%, 84%, and 88% compared to both PABFD and E-eco for synthetic trace1, trace2, and trace3, respectively.

As expected from the time complexity analysis of the proposed, PABFD, and E-eco, their runtime goes slightly higher when the number of VMs and hosts of the data center are increased.

6.7 Performance Degradation Detection

Providing labeled data resulting from server and VM monitoring to assess the performance variation, originating from various anomalies is not practical. Therefore, here we use a constant value instead of Equation (13) for the parameter α, which causes power mode selection agents to learn constantly as they act. In this way, we can investigate the ability of MAGNETIC in detecting performance degradation. This parameter has been tuned based on the objectives and chosen to be 0.6 for this simulations.

TABLE 6: Synthetic workload traces

	Number of VMs	Number of hosts
Synthetic trace1	3877	4000
Synthetic trace2	5108	5200
Synthetic trace3	7522	7600

TABLE 7: Energy consumption (kWh) and average SLA violation (%) of different algorithms for synthetic traces

	NPA		PABFD		E-eco		MAGNETIC	
	Energy Consumption	SLA Violation	Energy Consumption	SLA Violation	Energy Consumption	SLA Violation	Energy Consumption	SLA Violation
Synthetic trace1	3848.63	0	1805.21	12.07	1921.48	12.61	1640.23	12.78
Synthetic trace2	5272.23	0	2485.41	11.81	2651.06	12.73	2257.85	12.84
Synthetic trace3	7496.16	0	3507.20	12.10	3735.35	12.88	3190	12.5

TABLE 8: Runtime (seconds) and number of migrations (#) of different algorithms for synthetic traces

	NPA		PABFD		E-eco		MAGNETIC	
	RunTime	Migrations	RunTime	Migrations	RunTime	Migrations	RunTime	Migrations
Synthetic trace1	0.15	0	4.03	98464	3.11	98428	8.71	16605
Synthetic trace2	0.2	0	6.25	134981	4.77	135031	18.75	20984
Synthetic trace3	0.39	0	12.15	192078	9.09	192223	25.2	21202

As a proof of concept, we have simulated a data center with 900 hosts for the 03/06 workload trace (see Section 6.1) in which 5 hosts suddenly experience a 50% performance degradation, that is revealed in C^i_{Alloc} of the degraded hosts. The effect of the severity of degradation is investigated latter in this section. As after initial iterations around 50 hosts are in *Active* mode, around 10% of the hosts are being degraded.

The results of the initial evaluation are shown in Fig. 8. In this evaluation, the performance degradation has started after four days of simulation. As shown in this figure, the selected power mode for all degraded hosts is chosen to be *DSleep*, after several iterations (Y-axis represents the percentage of the total number of history entries of degraded hosts, in each power mode). The number of iterations required to drain these hosts from VMs depends on the saved Q values and VMs placement of other hosts. Also, Fig. 9 represents number of running and migrating VMs of the five degraded hosts during one week for several scenarios. These scenarios are different concerning the severity of applied degradation. Performance degradation has not been applied in Fig. 9 (a) and (b) and the results in these two sub-figures are different in their α parameter (we present these two sub-figures to show number of running and migrating VMs under normal operations). As we can see from Fig. 9 (c) and (d), after applying performance degradation (end of the fourth day), the number of migrating VMs increases, which leads to decreasing number of running VMs on degraded hosts. Also, as the severity of degradation increases, MAGNETIC increases the number of migrating VMs to drive degraded hosts into the *DSleep* power mode in fewer iterations.

To evaluate the effect of the severity of performance degradation, we have simulated various levels of degradation (10%-90%) for one active host and repeated the simulations for each degradation level five times. Fig. 10 shows the average number of needed iterations for the proposed method to switch a degraded host into *Sleep*/*DSleep* mode, after observing the different levels of degradation. As we can see from this figure, the proposed method drives degraded hosts with a higher degree of degradation, into *Sleep*/*DSleep* modes in fewer iterations. For example, the proposed approach detects 50% and 90% of performance degradation in 100 and 16 iterations respectively which corresponds to about 8 and 1 hours respectively.

In summary, MAGNETIC is able to observe the faults that have an effect on the host's energy consumption or its allocated MIPS. For example, as a host power failure causes the associated agent to receive a reward composed of zero energy, zero allocated MIPS, and some requested MIPS, the failure can be observed by the agent while detecting network degradation needs further heuristics or machine learning based algorithms.

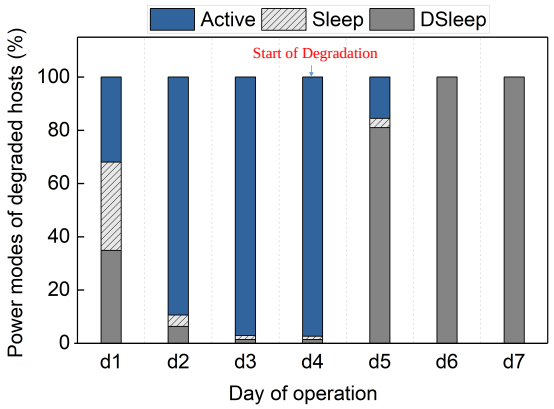

Fig. 8: Power modes of degraded hosts during one week

Finally, the capability of our solution in detecting the faults (which have an effect on the host's energy consumption or its allocated MIPS), in addition to the type and severity of the fault's effects depends on the learning parameters. In particular, the capability of our system in detecting intermittent faults depends on the period of the fault and on the parameters α and ε. The higher the ε the higher the chance of activating a host which has been driven in a sleep mode (because of the intermittent fault) in the previous time slots.

7 Conclusion

In this paper, an approach has been proposed to improve energy and resource efficiency of the data centers. To achieve this objective, a learning-based power mode selection unit and a migration unit have been designed to manage physical hosts and VMs of the data center simultaneously. The proposed MAGNETIC scheme is a centralized-distributed approach which considers the overheads associated with PMT and VM migration in its policy. The

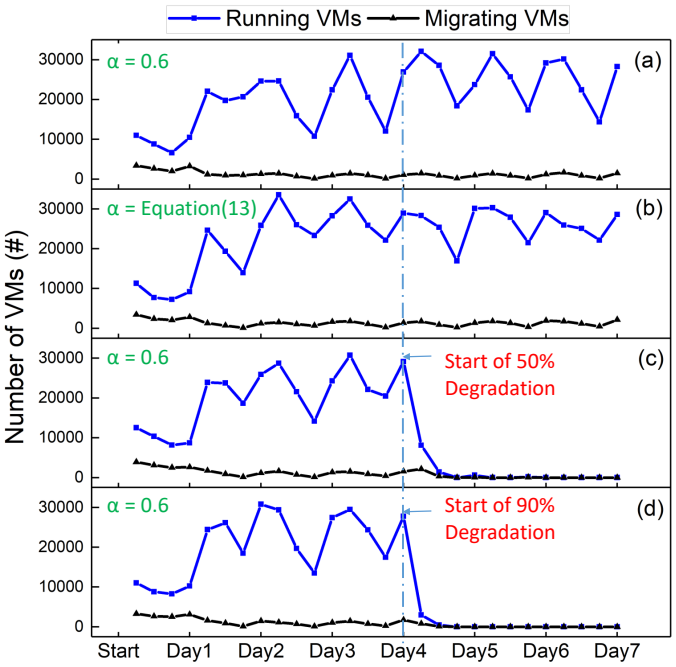

Fig. 9: Number of running and migrating VMs of degraded hosts during one week. (a) $\alpha = 0.6$ and no performance degradation (b) $\alpha =$ Equation(13) and no performance degradation (c) $\alpha = 0.6$ and 50% of performance degradation (d) $\alpha = 0.6$ and 90% of performance degradation

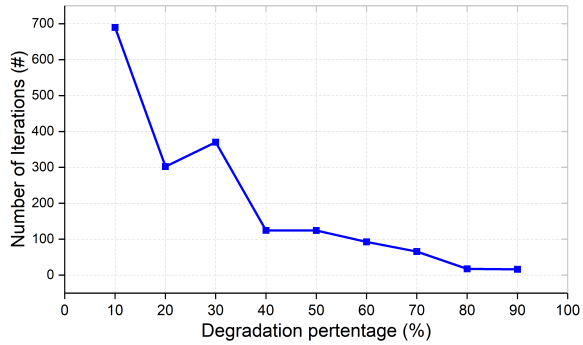

Fig. 10: Average number of needed iterations for MAGNETIC to switch a degraded host into $Sleep/DSleep$ mode

proposed scheme has been evaluated using PlanetLab workload traces in terms of energy consumption, average SLA violation, and runtime. The results show that the proposed approach decreases energy consumption when compared to approaches in the current SoA while its average SLA violation is almost the same. Our obtained results also show that the proposed approach uses minimum number of VM migrations and PMTs with respect to other approaches. Moreover, the algorithm keeps its efficiency for higher scale benchmarks and data centers, which proves its scalability. In future work, we envision to apply our proposed approach in heterogeneous or hybrid CPU/GPGPU data centers, for parallel applications.

ACKNOWLEDGMENTS

This work has been partially supported by the EC H2020 MANGO (GA No. 671668) project, the EC H2020 DeepHealth (GA No. 825111) project, and the ERC Consolidator Grant COMPUSAPIEN (GA No. 725657). Also funding from Ministry of Science, Research and Technology of Islamic Republic of Iran is gratefully acknowledged.

REFERENCES

[1] M. Zapater *et al.*, "Self-organizing maps versus growing neural gas in detecting anomalies in data centres," *Logic Journal of the IGPL*, vol. 23, no. 3, pp. 495–505, 2015.
[2] J. Koomey, "Growth in data center electricity use 2005 to 2010," *A report by Analytical Press, completed at the request of The New York Times*, vol. 9, 2011.
[3] M. Avgerinou *et al.*, "Trends in data centre energy consumption under the european code of conduct for data centre energy efficiency," *Energies*, vol. 10, no. 10, p. 1470, 2017.
[4] A. Shehabi *et al.*, "United states data center energy usage report," 2016.
[5] O. Tuncer *et al.*, "Diagnosing performance variations in hpc applications using machine learning," in *High Performance Computing*, J. M. Kunkel *et al.*, Eds. Springer International Publishing, 2017, pp. 355–373.
[6] X. Wang and J. R. McDonald, *Modern power system planning*. McGraw-Hill Companies, 1994.
[7] A. Pahlevan *et al.*, "Energy proportionality in near-threshold computing servers and cloud data centers: Consolidating or not?" in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018.
[8] M. Abdullah *et al.*, "A heuristic-based approach for dynamic vms consolidation in cloud data centers," *Arabian Journal for Science and Engineering*, vol. 42, no. 8, pp. 3535–3549, 2017.
[9] N. J. Kansal and I. Chana, "Energy-aware virtual machine migration for cloud computing-a firefly optimization approach," *Journal of Grid Computing*, vol. 14, no. 2, pp. 327–345, 2016.
[10] 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.
[11] T. H. Nguyen *et al.*, "Virtual machine consolidation with multiple usage prediction for energy-efficient cloud data centers," *IEEE Transactions on Services Computing*, 2017.
[12] A. Verma *et al.*, "pmapper: power and migration cost aware application placement in virtualized systems," in *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*. Springer-Verlag New York, Inc., 2008, pp. 243–264.
[13] 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.
[14] Q. Wu *et al.*, "Energy and migration cost-aware dynamic virtual machine consolidation in heterogeneous cloud datacenters," *IEEE TSC*, 2016.
[15] F. D. Rossi *et al.*, "E-eco: Performance-aware energy-efficient cloud data center orchestration," *Journal of Network and Computer Applications*, vol. 78, pp. 83–96, 2017.
[16] C. Isci *et al.*, "Agile, efficient virtualization power management with low-latency server power states," in *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3. ACM, 2013, pp. 96–107.
[17] A. Agelastos *et al.*, "The lightweight distributed metric service: a scalable infrastructure for continuous monitoring of large scale computing systems and applications," in *International Conference on High Performance Computing, Networking, Storage and Analysis*, 2014, pp. 154–165.
[18] A. Varasteh and M. Goudarzi, "Server consolidation techniques in virtualized data centers: A survey," *IEEE Systems Journal*, vol. 11, no. 2, pp. 772–783, 2017.
[19] C. Watkins and P. Dayan, "gtechnical note: Q-learning, h machine learning, vol. 8," 1992.
[20] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, 1989.
[21] J. Hu and M. P. Wellman, "Nash q-learning for general-sum stochastic games," *Journal of machine learning research*, vol. 4, no. Nov, pp. 1039–1069, 2003.
[22] R. N. Calheiros *et al.*, "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.
[23] E. Feller *et al.*, "A case for fully decentralized dynamic vm consolidation in clouds," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*. IEEE, 2012, pp. 26–33.

[24] S. Voulgaris *et al.*, "Cyclon: Inexpensive membership management for unstructured p2p overlays," *Journal of Network and Systems Management*, vol. 13, no. 2, pp. 197–217, 2005.
[25] J. A. Brandt *et al.*, "Enabling advanced operational analysis through multi-subsystem data integration on trinity," in *Proc. Cray Users Group*, 2015.
[26] A. Agelastos *et al.*, "Toward rapid understanding of production hpc applications and systems," in *IEEE International Conference on Cluster Computing*, 2015, pp. 464–473.
[27] J. Xu *et al.*, "Survivable virtual infrastructure mapping in virtualized data centers," in *IEEE International Conference on Cloud Computing*, 2012, pp. 196–203.
[28] E. Baseman *et al.*, "Interpretable anomaly detection for monitoring of high performance computing systems," in *Outlier Definition, Detection, and Description on Demand Workshop at ACM SIGKDD*, 2016.
[29] O. Ibidunmoye, "Performance anomaly detection and resolution for autonomous clouds," Ph.D. dissertation, Umeå University, 2017.
[30] A. Beloglazov *et al.*, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.
[31] M. Zapater *et al.*, "Leakage-aware cooling management for improving server energy efficiency," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 26, no. 10, pp. 2764–2777, 2015.
[32] H. Chen *et al.*, "The data center as a grid load stabilizer," in *19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2014, pp. 105–112.
[33] ——, "Dynamic server power capping for enabling data center participation in power markets," in *Proceedings of the International Conference on Computer-Aided Design*, ser. ICCAD '13. IEEE, 2013, pp. 122–129.
[34] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Princeton University, 2011.
[35] H. Liu *et al.*, "Performance and energy modeling for live migration of virtual machines," *Cluster Computing*, vol. 16, no. 2, pp. 249–264, 2013.
[36] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.
[37] L. P. Kaelbling *et al.*, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.
[38] M. Tokic, "Adaptive ε-greedy exploration in reinforcement learning based on value differences," in *Annual Conference on Artificial Intelligence*. Springer, 2010, pp. 203–210.
[39] K. Park and V. S. Pai, "Comon: a mostly-scalable monitoring system for planetlab," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 1, pp. 65–74, 2006.
[40] A. M. Devices, "Amd family 10h server and workstation processor power and thermal data sheet," https://support.amd.com/TechDocs/43374.pdf, 2010.

Kawsar Haghshenas received the BS degree from K.N.Toosi University of Technology in 2012, and the MS degree from Sharif University of Technology in 2014, both in computer engineering. She is currently working toward the PhD degree at the Dependable Systems Design Laboratory, Electrical and Computer Engineering Department, University of Tehran. She was working in the Embedded Systems Laboratory at EPFL University from 2017 to 2018. Her research interests include VLSI circuits, energy optimization, and cloud computing.

Ali Pahlevan is currently a Ph.D. candidate in Electrical Engineering (EE) in the Embedded Systems Laboratory (ESL) at Swiss Federal Institute of Technology Lausanne (EPFL). He received his M.Sc. degree in Computer Engineering from Sharif University of Technology in 2012, and B.Sc. degree in Computer Engineering from Ferdowsi University of Mashhad in 2010. His research interests focus on system-level energy optimization techniques in the area of data centers and cloud computing.

Marina Zapater is currently is a Post-Doctoral researcher in the ESL at EPFL. She was Visiting Assistant Professor in the Computer Architecture Department at Universidad Complutense de Madrid, Spain, in the academic year 2015-2016. She received her Ph.D. degree in Electronic Engineering from Universidad Politecnica de Madrid in 2015, an M.Sc. in Telecommunication Engineering degree and a M.Sc. in Electronic Engineering degree, both from the Universitat Politecnica de Catalunya, in 2010. Her research interests include proactive and reactive thermal and power optimization of complex heterogeneous systems, energy efficiency in data centers, ultra-low power architectures and embedded systems. In this area, she has co-authored over 25 publications in top-notch international conferences and journals, and she has participated in several national and international research projects. She is a member of IEEE and CEDA and has served as TPC member of several conferences, including VLSI-SoC and MCSoC.

Siamak Mohammadi received his B.Sc., M.Sc. and Ph.D. degrees from the University of Paris Sud Orsay, France in 1990, 1992 and 1996, respectively, all in electrical engineering. During his PhD he was supported by a grant from the Ministry of Education of France. From 1997 to 1999 he was a Research Associate with the Department of Computer Science, University of Manchester, England. In 1999 he moved to Canada and worked at Cogency Semiconductor Inc. in Toronto until 2003 and then at ATI Technologies Inc. until 2005. Currently he is an Associate Professor in School of Electrical and Computer engineering, at the University of Tehran, Iran. He has over 15 years experience in VLSI digital area, ASIC design and verification, asynchronous design, and on-chip interconnects in GALS NoCs. He has contributed to the design of the first asynchronous ARM microprocessor, as well as several PowerLine networking and RFID chips.

David Atienza (M'05-SM'13-F'16) received his M.Sc. and Ph.D. degrees in Computer Science and Engineering from Complutense Univ. of Madrid (UCM), Spain, and IMEC, Belgium, in 2001 and 2005. Currently he is an associate professor of electrical and computer engineering, and head of the Embedded Systems Laboratory (ESL) at the Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland. His research interests include system-level design methodologies for high-performance multi-processor system-on-chip (MPSoC) and low-power Internet-of-Things (IoT) systems, including new 2-D/3-D thermal-aware design for MPSoCs and many-core servers, ultra-low power system architectures for wireless body sensor nodes and edge computing, HW/SW reconfigurable systems, dynamic memory optimizations, and network-on-chip design. He is a co-author of more than 250 publications in peer-reviewed international journals and conferences, several book chapters, and eight U.S. patents in these fields. He has earned several best paper awards and he is (or has been) an Associate Editor of IEEE TC, IEEE D&T, IEEE T-TCAD, IEEE T-SUSC and Elsevier *Integration*. He was the Technical Programme Chair of IEEE/ACM DATE 2015 and General Programme Chair of IEEE/ACM DATE 2017. Dr. Atienza received the DAC Under-40 Innovators Award in 2018, the IEEE TCCPS Mid-Career Award in 2018, an ERC Consolidator Grant in 2016, the IEEE CEDA Early Career Award in 2013, the ACM SIGDA Outstanding New Faculty Award in 2012, and a Faculty Award from Sun Labs at Oracle in 2011. He is an IEEE Fellow and an ACM Distinguished Member.