2014 IEEE 6th International Conference on Cloud Computing Technology and Science

Hierarchical VM Management Architecture for Cloud Data Centers

Fahimeh Farahnakian, Pasi Liljeberg, Tapio Pahikkala, Juha Plosila, Hannu Tenhunen
Department of Information Technology, University of Turku, Turku, Finland.
fahime.farahnakian@utu.fi, pasi.liljeberg@utu.fi, tapio.pahikkala@utu.fi, juha.plosila@utu.fi, hannu.tenhunen@utu.fi

***Abstract*—Efficient energy use has become a critical issue for designing and managing of cloud data centers. Virtualization is a key technology for reducing energy cost and improving resource utilization in data centers. One of the challenges faced by virtualized data centers is to decide how to pack VMs on the least number of physical machines. This paper presents a VM management framework which is based on a multi-agent system to minimize energy consumption and Service Level Agreement (SLA) violations. The proposed agents are arranged in a three-level hierarchical structure to perform VM assignment, VM placement and VM consolidation in a data center efficiently. Experimental results demonstrate that the framework achieves high quality solution in spite of its simplicity and scalability.**

***Keywords*-Dynamic VM consolidation; energy-efficiency; Hierarchical model; cloud computing; green data center; SLA**

I. Introduction

Energy management is becoming an increasingly important topic for the cloud data centers, due to the high operational expenditure in the form of power consumption as well as the concerns for global warming. Dynamic Virtual Machine Consolidation (DVMC) has a major impact on energy efficiency by packing Virtual Machines (VMs) into the minimum number of servers and switching the idle servers to the power-saving state [1]. DVMC leverages virtualization technology such as Xen [2] and VmWare [3] to increase the server utilization and power efficiency of data centers. This technology allows a Physical Machine (PM) to be shared between multiple through a VM monitor or hypervisor, where each VM can run multiple application tasks. In addition, virtualization provides the ability to transfer a VM between PMs without suspension and with minimal service downtime using live migration [4]. However, achieving the desired level of Quality of Service (QoS) between users and a data center is critical. The QoS requirements are formalized via Service Level Agreement (SLA) that describes characteristics such as minimal throughput, maximal response time or latency delivered by the deployed system.

Most of the centralized VM management architectures are not scalable to control VMs in a large-scale data center for different reasons. First, the worst-case computational complexity of a centralized controller is commonly proportional to the system size and thus cannot scale well for large-scale systems. Since every server in the data center may need to communicate with the centralized controller in every control period, the controller may become a communication bottleneck. Furthermore, a centralized controller may have long communication delays in large-scale systems. To address these problems, we present a hierarchical architecture that has the ability to scale for thousands of PMs. We propose three types of agents to control the system in a hierarchical manner: global, cluster and local agents. The term *agent* originally comes from Artificial Intelligence (AI) and refers to anything that can be viewed by its environment through sensors and acting upon the environment by actuators. In this paper, the term agents refers to software agents. The proposed agents cooperate together to manage VMs in order to reduce the energy consumption and SLA violations in a large-scale data center. The simulation results show that the proposed architecture maintains the desired QoS while reducing the energy consumption in a data center.

The remainder of the paper is organized as follows. Section II surveys some literature regarding to dynamic consolidation in cloud environment. Section III presents the system architecture. The proposed mechanisms and algorithms that are used by agents to perform VM consolidation are described in Section IV. Section V shows the implementation of the proposed method. Finally, we give the experimental results and conclusion in Section VI and VII.

II. Related Work

In recent years, significant research has been done to reduce the energy cost in the cloud data centers. Much of the prior work has attempted to reduce energy consumption by consolidating VMs into a minimum number of active PMs. Note that active means the PM has been assigned at least one VM. The main idea in the existing VM consolidation approaches such as [5], [6] is to use live migration to consolidate VMs periodically. In some approaches, VM consolidation has formulated as an optimization problem [5], [7]. Although an optimization problem is associated with constraints like data center capacity and SLA. Therefore, these works utilize a heuristic method for the multi-dimensional bin packing problem as an algorithm for the VM consolidation. In [5] a server consolidation algorithm, Secron, is proposed to minimize the overall number of used servers and the number of migrations. Sercon migrates VMs from the least loaded nodes to most loaded ones, respecting number of constraints described in the problem modeling, so that least loaded nodes can be released.

One way to improve the system scalability is to utilize a hierarchical architecture. Only a few works have investigated the use of hierarchical architectures for VM management. Mistral [8] is made of multiple controllers, the so-called Mistral controllers which of them managing a set of PMs. The authors argue that Mistral can be organized in a hierarchical manner to allow the management of large-scale systems. However, only small scale experiments on 8 PMs and 20 VMs are conducted to demonstrate the viability of the system. In [9] the authors introduce Eucalyptus which is an open source cloud computing framework focused on academic research. The Eucalyptus architecture is hierarchical and composes of four high level components for user-controlled virtual machine creation and existing resources control in a hierarchical fashion. In [10] a hierarchical resource management solution is proposed that used two managers. A global manager first assigns VMs into a cluster, then a local manager deploys the VMs to PMs in the cluster. The simulation results for a large number of VMs show the proposed scheme is scalable. Snooze [11] is a Infrastructure-as-a-Service (IaaS) cloud management system, which is designed to scale across many thousands of servers and VMs while being easy to configure, highly available, and energy efficient. For scalability, Snooze performs distributed VM management based on a hierarchical architecture. Distributed VM management is achieved by splitting the data center into independently managed groups of PMs and VMs.

We proposed the centralized and distributed architectures to manage the power and performance of data center in our previous works [12], [13], [14]. In [12], there is a central global controller to optimizes VM placement based on a power mode detection policy. The policy uses reinforcement learning method to detect the appropriate power mode of PM (active or sleep) according to the resource utilization. In order to speed up the learning time and address the mentioned problems of a centralized architecture, we presented a distributed architecture in [14]. The distributed architecture consist of several local controllers to detect PMs status that each local controller only observes its own state. Moreover, we used the ant colony system [13] to find a near-optimal VM placement solution based on the specified objective function by a global controller. In this paper, we extend the proposed distributed system model in our prior works and make new contributions by considering a hierarchical three-level model, and enable significant power saving with minimum migrations and SLA violations. Moreover, the hierarchical model provides the ability for efficient VM assignment, VM placement and VM consolidation. Generally, we develop a scalable, three-level hierarchical controller that can deal with a large number of PMs and VMs.

III. System Architecture

We assume a large-scale data center as a resource provider that consists of m heterogeneous Physical Machines (PMs). Each PM has a processor, which can be multi-core, with performance defined in Millions Instructions Per Second (MIPS). Besides that, a PM is characterized by the amount of RAM, network bandwidth and storage. Several users submit requests for provisioning of n VMs which are allocated to the PMs. The length of each request is specified by millions of instructions (MI).

In this paper, we propose a hierarchical architecture based on a three-tier data center topology to control VMs in order to reduce SLA violations and energy consumption in the data center. In this conventional topology [15], the Top-of-Rack (ToR) switch in the access layer provides connectivity to the PMs placed on every cluster. Each aggregation switch (AS) in the aggregation layer (sometimes referred to as distribution layer) forwards traffic from multiple access layer (ToR) switches to the core layer. Every ToR switch is connected to multiple aggregation switches for redundancy. The core layer provides secure connectivity between aggregation switches and core routers connected to the Internet. Figure 1 depicts an example of the Hierarchical VM management (HiVM) architecture that is made up of three types of agents: local, cluster and global agent. The local agent as a software module in each PM is responsible to control the resource utilization of the PM. Modern data centers consist of many clusters, and each cluster is a logical group of multiple PMs. The number of PMs in a cluster is bounded because the cluster's resource capacity is limited. Therefore, we propose a Cluster Agent (CA) to manage cluster's resources and control a set of LAs. Moreover, there is a Global Agent (GA) as a master node to manage CAs. So GA has an overall view of all clusters in the data center. Generally, these agents cooperate with each other to control available resources efficiently. The key idea of the HiVM architecture is to split the large VM management problem into a number of small problems such as VM assignment, VM placement and VM consolidation.

IV. Energy-Aware Hierarchy VM Management

In this section, we present mechanisms and algorithms which are used by the proposed agents in the HiVM system model. The main mechanisms include VM assignment, VM placement and VM consolidation.

A. Global Agent (GA)

The GA first receives the used and available CPU capacity information of all clusters in the data center through CAs. The used capacity of the cluster c represents the aggregated of used capacity of all PMs as

$$U_c = \sum_{i=1}^{k} U_i \tag{1}$$

where k is the number of PMs in the cluster c and U_i indicates the used CPU capacity of each PM that resides in the cluster. In the similar way, the available CPU capacity of the cluster, C_c, indicates the total available capacity of all PMs in the cluster. The GA is responsible to assign VMs to CAs based on clusters information. The VMs are sorted in decreasing order of their requested utilization. The clusters are also sorted in decreasing order of their used capacity. Then, the algorithm starts with the first VM and first cluster. If the cluster has enough capacity for the requested utilization by the VM, it assigns the VM to the cluster and considers

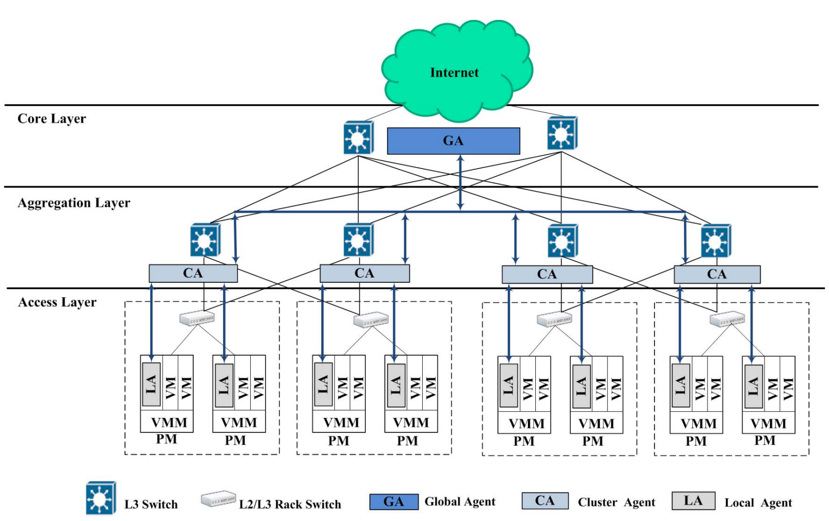

Fig. 1. An example of the HiVM architecture

next VM. Otherwise the algorithm considers next cluster for assigning the first VM. This constraint should hold in order to avoid performance degradation. The objective is to assign VMs to clusters so as to minimize the total amount of allocated resource while allocating enough resources to each VM.

B. Cluster Agents (CAs)

Each CA is in charge of the resource management of a set of PMs in a cluster. It has connectivity to the local and global agents. The CA receives PMs' status data such as available capacity, used capacity and predicted CPU utilization and PMs' status data from LAs. It performs VM placement and VM consolidation based on the received information. The VM placement algorithm is triggered once VM submission requests arrive from the GA. It allocates VMs to PMs based on the Best Fit Decreasing (BFD) algorithm as a best known heuristic for bin-packing problem. BFD sorts all VMs by utilization weights in decreasing order before being allocate to PMs. Therefore, it starts with VMs that require the largest amount of CPU utilization. The BFD algorithm allocate a VM to a PM so that after allocation the PM has a minimal space left over. It selects the PM for which the amount of available resources is closest to the requested amount of resources by the VM.

Since the VMs experience dynamic workloads, the CPU usage by a VM arbitrarily varies over time. So, a dynamic consolidation mechanism can optimize VM placement based on the current requested resource utilization of VMs. It performs several live VM migrations to reach such a state where minimal amount of PMs are used. The CA performs the VM consolidation algorithm (Algorithm 1) periodically based on the PMs' status of a cluster in two phases.

At the first phase (lines 3–9), the CA aims to migrate some VMs from the overloaded PMs (P_{over}) and predicted overloaded PMs ($\hat{P}_{over}$) to other PMs in order to reduce SLA violations. Therefore, it uses a VM selection policy to select

Algorithm 1 VM consolidation

```
1: $M = \emptyset$
2: $P = [P_{over}, \hat{P}_{over}]$
3: for $p \in P$ do
4:     apply VM selection policy to select $V_m$ of $V_p$
5:     for $v \in V_m$ do
6:         apply the VM allocation policy to select $p_{de}$
7:         $M = M \cup \{(p, v, p_{de})\}$
8:     end for
9: end for
10: for $p \in P_{under}$ do
11:     for $v \in V_p$ do
12:         apply the VM allocation to select $p_{de}$
13:         $M = M \cup \{(p, v, p_{de})\}$
14:     end for
15: end for
```

VMs for migration (line 4). As the duration of the migration process depends on the memory used by the VM, the proposed policy selects a VM that requires minimum memory. After a VM selection, the PM is checked again for being overloaded. If it is still considered being overloaded, the VM selection policy is applied again to select another VM to migrate from the PM. This is repeated until the PM is considered being not overloaded. In order to find the destination PM p_{de} for reallocating the migrated VMs (V_m), the CA runs Algorithm 2.

At the second phase (lines 10–15), the consolidation algorithm tries to migrate all VMs from the PMs with the minimum utilization (P_{under}) to other PMs if there is some destination PMs with enough capacity to reallocate VMs. In case of all VMs from an under-loaded PMs are migrated to other PMs, then the idle PMs switch to the sleep mode in order to reduce the energy consumption in the data center. The output of the VM consolidation algorithm is a migration plan that is sent to LCs to manage VM migrations. The migration plan is a set of 3-tuple ($p_{so}; v; p_{de}$), where the VM to be migrated is v, the source PM is p_{so} and the destination PM is p_{de}. If CA does not find a destination PM for allocating the migrated VM in the same cluster, it sends the request to GA for choosing p_{de} in another cluster.

Algorithm 2 VM allocation policy

```
1: $p_{de} = \emptyset$
2: for $p \in sortedPMList$ do
3:     if $(U_p + U_v \leq C_p)$ & $(PU_p + U_v \leq C_p)$ then
4:         $p_{de} = p$
5:         break
6:     end if
7: end for
8: return $p_{de}$
```

The proposed VM allocation policy (Algorithm 2) first sorts PMs in decreasing order according to the predicted CPU utilization data. Then, it selects first PM that has enough free resources to host the VM at the moment and near future. Therefore, the destination PM does not become overloaded after allocation by predicting on utilization of PM. Moreover, this policy can reduce the number of migrations by choosing the most loaded PM as a candidate for p_{de} instead of the least loaded PM. PU_p and U_p present the predicted and current CPU usage of PM p, respectively. U_v is the used CPU capacity of VM v and C_p is the available CPU capacity of PM p.

C. Local Agents (LAs)

The role of each LA is to determine the PM 's status based on the following conditions:

- If the current CPU utilization exceeds the available CPU capacity, the PM is considered as a member of P_{over}.
- If the predicted CPU utilization exceeds the available CPU capacity, the PM is considered as a member of $\hat{P}_{over}$.

In order to predict the utilization of a PM, the LA utilizes the LiRCUP [16]. LiRCUP uses the historical usage data during the life time of PM and approximates a function. Then, it predicts the utilization of a PM based on the current usage and the approximated function. Moreover, the LA controls VMs (e.g. start, stop, migrate) based on the received migration plan from its respective CA. It sends the VM live migration commands to the VM Monitor (VMM) to enforce the actual migration of VMs based on the received commands.

V. Experimental Setup

Implementation of our architecture has been performed on the CloudSim toolkit [17]. CloudSim is flexible, scalable and efficient for VM management techniques and provisioning policies. We simulated a data center comprising 800 heterogeneous PMs arranged into 40 clusters where each cluster holds 20 servers. The tree-tier data center topology consist of 8 core, 16 aggregation and 40 access switches. We have 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 VM management algorithms designed for multi-core CPU architectures [18]. In addition, we consider four essential metric to show the efficiency of the proposed architecture.

A. SLA Violations

This metric (SLAV) [18] combines the SLA violation due to over-utilization (SLAVO) and SLA violation due to migration (SLAVM) as

$$SLAV = SLAVO \times SLAVM \tag{2}$$

$$SLAVO = \frac{1}{M}\sum_{i=1}^{M}\frac{T_{s_i}}{T_{a_i}} \tag{3}$$

where M is the number of PMs; T_{s_i} is the total time that the PM i has experienced utilization of 100% leading to an SLA violation. T_{a_i} is the total time of the PM i being the active state.

$$SLAVM = \frac{1}{N}\sum_{j=1}^{N}\frac{C_{d_j}}{C_{r_j}} \tag{4}$$

where N is the number of VMs; C_{d_j} is the estimate of the performance degradation of the VM j caused by migrations; C_{r_j} is the total CPU capacity requested by the VM j during its lifetime. In our experiments, we estimate C_{d_j} as 10% of the CPU utilization in MIPS during all migrations of the VM j.

B. Energy Consumption

This metric indicates the total energy consumption of the physical resources of a data center caused by the different application workloads. Table I illustrates the real power consumption characteristics of the selected servers in the simulator. Therefore, instead of using an analytical model of power consumption by a server, we utilize real data on power consumption provided by the results of the SPECpower benchmark[1].

C. Number of VM Migrations

Live migration is a costly operation that include some amount of CPU processing during the migration PM, the link bandwidth between the migrating and migrated PMs, downtime of the services on the migrating VM and total migration time [5]. Therefore, it is crucial to minimize the number of VM migrations. The length of a live migration depends on the total amount of memory used by the VM and available network bandwidth. In our simulation, all links are 1Gbps network links.

D. Energy and SLA Violations

The objective of the proposed architecture is to minimize both energy and SLA violations. Therefore, we propose a combined metric that captures both energy consumption (EC) and the level of SLA violations (SLAV), which we denote Energy and SLA Violations (ESV) as

$$ESV = EC \times SLAV \tag{5}$$

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

TABLE I
THE POWER CONSUMPTION AT DIFFERENT LOAD LEVELS IN WATTS

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

VI. EXPERIMENTAL RESULTS

We compare the experimental results of the proposed Hierarchical VM management (HiVM) method with a hierarchical architecture presented in [18] that consist of local and global managers. The local manager as a module of the VMM monitors the PM's CPU utilization and decides when and which VMs should be migrated. The global manager resides on the master node and collects information from the local managers to maintain the overall view of the utilization of resources. The local managers use three algorithms to set adaptive upper and lower utilization thresholds and keep the total CPU utilization of a PM between these bounds. When the upper bound is exceeded, VMs are reallocated for load balancing and when the utilization of a PM drops below the lower bound, VMs are reallocated for consolidation. The algorithms [18] adapt the utilization threshold dynamically based on the Median Absolute Deviation (MAD), the Interquartile Range (IQR) and Local Regression (LoR) approaches. In addition, we consider the static threshold method (THR) in [18] 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 type of workloads: random and real.

A. *Random Workload*

In the random workload, the users submit 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 is modeled to generate the utilization of CPU according to a uniformly distributed random variable. The application runs for 150,000 MI that is equal to 10 minutes of the execution on 250 MIPS CPU with 100% utilization.

Figure 2 illustrates the SLAV, energy consumption, number of VM migrations and ESV metric caused by the Hi-VM, THR, MAD, IQR and LoR methods in the random workload. Our proposed hierarchical VM consolidation can reduce the SLA violations rate more efficiently than other techniques (Figure 2(a)). The obtained results can be explained by the fact that the proposed HiVM minimizes the amount of SLA violations by migrating VMs from the overloaded and predicted overloaded PMs. Moreover, the proposed VM allocation policy guarantees the destination PM for allocating migrated VM is not overloaded at the moment and near future.

In addition, Figure 2(b) shows HiVM can provide higher energy saving in comparison to other policies. It reduces the energy consumption up to 45.1% in the random workload. Figure 2(c) shows the total number of VMs migration during the VM consolidation. The proposed VM allocation and VM consolidation algorithms can decrease the number of VM migrations due to considering the predicted utilization of PMs. The results produced by the selected algorithms and HiVM for ESV metric are shown Figure 2(d).

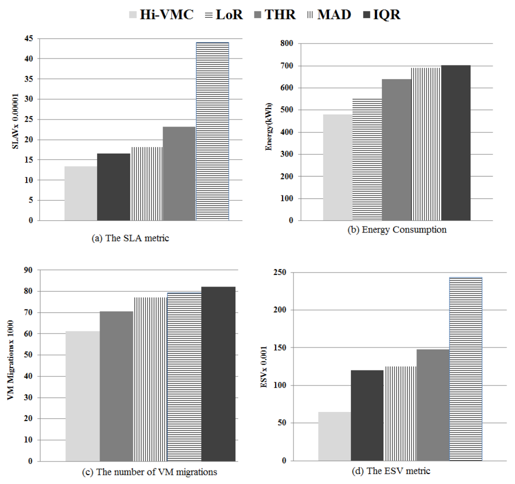

Fig. 2. The SLAV metric, energy consumption, number of VM migrations and ESV metric by HiVM and benchmark methods in the random workload

B. *Real Workload*

To show the proposed hierarchical architecture is scalable or not, we run the simulation for the real workload with large number of VMs. 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 [19]. 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 EC2, where VMs are created and managed by several independent users and the infrastructure provider is not aware what particular applications are executing in the VMs. We used five days from the workload traces collected during March 2011 of the project. During the simulation, each VM is randomly assigned a workload trace from one of the VMs from the corresponding day.

The SLAV metric for the real workload are shown in Figure 3(a). The results show that HiVM leads to significantly less SLA violations than other four benchmark algorithms. In HiVM, a signicant reduction of the energy consumption on 3 March of 48.0%, 26.3%, 42.6% and 45.7% can be observed by comparing THR, LoR, MAD and IQR, respectively. Further-

TABLE II
THE 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

more, the proposed hierarchical model can reduce the number of VM migrations and ESV metric efficiently.

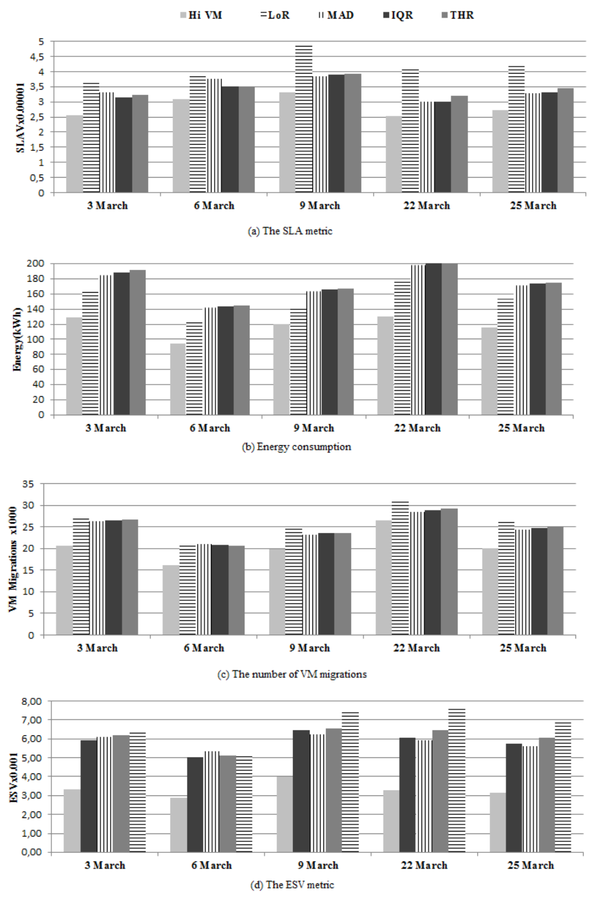

Fig. 3. The SLAV metric, energy consumption, number of VM migrations and ESV metric by HiVM and benchmark methods in the real workload

VII. CONCLUSION

In this paper, a three-level hierarchical architecture for VM management in a large-scale data center is proposed. This architecture uses multi agents to achieve the proposed objectives automatically. The objectives are to reduce the energy consumption and number of migrations in the data center while ensuring a high level of adherence to the SLA. Compared with the existing dynamic consolidation methods in CloudSim simulation, the proposed hierarchical VM consolidation model is able to reduce energy consumption, the number of migrations and SLA violations efficiently.

REFERENCES

[1] M. Pedram, "Energy-efficient datacenters," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 31, pp. 1465–1484, 2012.

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

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

[4] C. Clark, K. Fraser, S. Hand, J. 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 and Implementation*, vol. 2, 2005, pp. 273–286.

[5] 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.

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

[7] 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.

[8] G. Jung, M. Hiltunen, K. Joshi, R. Schlichting, and C. Pu, "Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures," in *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, 2010, pp. 62–73.

[9] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, 2009, pp. 124–131.

[10] I. Hwang and M. Pedram, "Hierarchical virtual machine consolidation in a cloud computing system," in *Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing*, 2013, pp. 196–203.

[11] E. Feller, L. Rilling, and C. Morin, "Snooze: A scalable and autonomic virtual machine management framework for private clouds," in *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, 2012, pp. 482–489.

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

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

[14] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, and H. Tenhunen, "Multi-agent based architecture for dynamic vm consolidation in cloud data centers," in *40th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, 2014, pp. 111–118.

[15] M. Bari, R. Boutaba, R. Esteves, L. Granville, M. Podlesny, M. Rabbani, Q. Zhang, and M. Zhani, "Data center network virtualization: A survey," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 2, pp. 909–928, Second 2013.

[16] 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 *39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, 2013, pp. 357–364.

[17] R. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience (SPE)*, vol. 41, pp. 23 – 50, 2011.

[18] A. Beloglazov and Buy, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience (CCPE)*, vol. 24, pp. 1397–1420, 2012.

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