Integrating VM Selection Criteria in Distributed Dynamic VM Consolidation Using Fuzzy Q-Learning

Seyed Saeid Masoumzadeh
Research Group Entertainment Computing
University of Vienna
Vienna, Austria
Email: Masoumzadeh@gmail.com

Helmut Hlavacs
Research Group Entertainment Computing
University of Vienna
Vienna, Austria
Email: Helmut.Hlavacs@univie.ac.at

***Abstract*—Distributed dynamic VM consolidation can be an effective strategy to improve energy efficiency in cloud environments. In general, this strategy can be decomposed into four decision-making tasks: (1) Host overloading detection, (2) VM selection, (3) Host underloading detection, and (4) VM placement. The goal is to consolidate virtual machines dynamically in a way that optimizes the energy-performance tradeoff online. In fact, this goal is achieved when each of the aforementioned decisions are made in an optimized fashion. In this paper we concentrate on the VM selection task and propose a Fuzzy Q-Learning (FQL) technique so as to make optimal decisions to select virtual machines for migration. We validate our approach with the CloudSim toolkit using real world PlanetLab workload. Experimental results show that using FQL yields far better results w.r.t. the energy-performance trade-off in cloud data centers in comparison to state of the art algorithms.**

***Index Terms*—Energy Efficient Cloud Data Center, Dynamic VM Consolidation, VM Selection, Fuzzy Q-Learning.**

I. Introduction

In the ICT world, data centers have a high proportion of the total energy consumption [1]. One of the ways to reduce energy consumption in data centers is to apply virtualization. The basic effect of virtualization is the ability to run fewer physical servers with higher per-server utilization as virtual machines (VMs), thus reducing the amount of the hardware in use. This is "server consolidation" has led to increased flexibility and availability of resources, while reducing hardware costs as well as energy consumption. Cloud computing is a new computing model based on data centers that leverages virtualization technology and provides on-demand resource provisioning over the Internet on a pay as you go basis. It is essential for cloud providers to offer Quality of Service (QoS) to their customers, being negotiated in terms of Service Level Agreements (SLA).

Unfortunately, server consolidation enabled by virtualization introduces a new problem to the cloud environment. Since the size of Virtual Machines (VMs) inside a physical server might change due to dynamic workloads, resource underutilization or overutilization might occur, causing either inefficient energy consumption or unwanted performance degradation. Consequently, the cloud managing system needs live migration of VMs to achieve server consolidation and maximized per-server utilization while attaining the promised non-functional qualities of the service guaranteed in the Service Level Agreements (SLA).

Distributed dynamic VM consolidation [2] can be an effective strategy to tackle this problem. The procedure of this strategy can be decomposed into four decision-making tasks: (1) Host overloading detection, here deciding when a host must be considered as overloaded. In this situation, one or more VMs must be migrated away from this host. (2) Host underloading detection, here deciding when a host must be considered to be underloaded. In this situation, the host is ready to switch to sleep mode and all VMs must be migrated away from this host. (3) VM selection, here deciding, which VMs should be migrated away from overloaded hosts, and (4) VM placement, here deciding about which host must be selected to receive migrating VMs. Indeed, the energy-performance trade-off optimization is achieved when each of aforementioned decisions are made dynamically in an optimized fashion.

Clouds are inherently dynamic due to their workload variability, and thus management decisions must be done dynamically too. So far, the research on dynamic VM consolidation procedure has focused mainly on host overloading detection [2] [3] [4]. VM selection, on the other hand, has been somehow neglected in the past, often using fixed criteria instead of dynamic ones. In this paper we consider the VM selection task as a Dynamic Decision-Making (DDM) task and model it using Fuzzy Q-Learning (FQL). We show how FQL can be used to dynamically choose VM selection strategies from a set of possible strategies in order to achieve better results than each of the individual strategies achieves when used alone.

The rest of the paper is organized as follows. In Section II we address the related work. In Section III we discuss our proposed approach. Section IV introduces Fuzzy Q-learning. In Section V we propose our system model. In Section VI we show how the VM selection task is formulated using FQL theory. In Sections VII and VIII we present the experimental setup and the results respectively.

II. Related Work

In this section we discuss prior approaches about VM selection in distributed dynamic VM consolidation. It is important to note that in previous works, the dynamic VM consolidation procedure has sometimes been called with other names such as dynamic resource allocation, or dynamic VM management. But the VM selection has always been part of the decision-making tasks in these procedures. Khanna et al. [5] defined a VM selection criterion based on CPU utilization of VMs. In fact, the VMs with the lowest CPU utilization are selected. Further, they proved that selecting VMs based on this criterion can result in minimized migration cost. Beloglazov et al. in [6] presented a criterion based on CPU capacity of VMs defined by VM parameters. According to this criterion, the selected VM has the lowest CPU capacity. They believe that decision-making based on this criterion can minimize the potential increase of the host's utilization and prevent SLA violations. Beloglazov et al. [2] also introduced three different criteria. The first one is based on migration time. A VM is selected if it requires the minimum time to complete a migration relatively to the other VMs allocated to the host. The migration time is estimated as the amount of RAM utilized by the VM, divided by the spare network bandwidth available to the host. The second criterion is based on CPU utilization and a selected VM has the highest correlation of the CPU utilization with other VMs. The idea of this criterion is that the higher the correlation between the resource usage by applications running on an oversubscribed server, the higher the probability of server overloading. All aforementioned approaches employ a fixed criterion for decision-making and as mentioned before are not suitable for decision-making in dynamic environments.

III. Integrating VM Selection Criteria

When a host is overloaded, the dynamic VM consolidation procedure calls the VM selection task in order to remedy this situation by migrating VMs. The VM selection task must choose VMs to migrate in a way that decreases performance degradation due to both overutilization in one hand and also migration on the other hand. Table I shows the result of dynamic VM consolidation procedure in simulation three days of cloud center management, and when using two different VM selection criteria. As shown the VM selection task that uses the *maximum utilization (MAXU)* criterion for selecting VM achieves a better result in SLA violations due to overutilization (SLAVO, less is better, see Section VI) than the *minimum utilization (MINU)* criterion. In fact, the maximum utilization criterion always chooses the VMs with the highest utilization. Consequently it can remedy performance degradation due to overutilization in less time. However, it decreases the number of migrations in comparison with the minimum utilization criterion, and in contrast, increases SLA violations due to migration (SLAVM, less is better, see Section VI), as the performance degradation due to migration depends on the CPU utilization of the VMs [5]. Apart from it, some aggressive behaviors of maximum utilization criterion might cause an underutilized situation and consequently more energy consumption (EC). This experiment shows the result of VM selection task is highly depended on the workload behavior and the state of the physical host. However, using one fixed criterion for selecting VMs might some times lead to satisfactory results, but in other situations to poor results. Therefore, in this situation an adaptive and predictive mechanism can be efficient to make decisions about which criterion can achieve the best result in the current state. In this paper we propose the FQL theory as an effective approach to create an adaptive and predictive VM selection during dynamic VM consolidation in cloud environments. In the following, we focus on modeling the VM selection task as a dynamic decision making task using FQL. We show how FQL as an intelligent decision maker can learn, how to use multiple criteria for selecting VMs in different states in order to optimize the energy-performance tradeoff dynamically.

TABLE I
Comparision Between Two Different Criteria

Criterion	SLAVM	SLAVO	EC (kWh)	Num. Migrations
MINU	1.26	4.22	257.49	94669
MAXU	1.31	3.92	265.87	29939

IV. Fuzzy Q-Learning

In standard reinforcement learning (RL) [7], at each time-step t, the agent as a learner interacts with the environment, observes the current state s_t of the environment, and selects an action a_t from the set of possible actions corresponding to the state. One time-step later, as a consequence of the chosen action, the agent receives a numerical reward r_{t+1}, and finds itself in a new state s_{t+1}. In a trial and error interaction, agents finally learn how to map states to actions, in order to maximize the discounted sum of rewards obtained, which is given by

$$\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}.$$

Here γ is a discount factor. This mapping denoted by π and is called agent control policy. The problem can be modeled by a Markov Decision Process (MDP) and solved using Dynamic Programming (DP). DP provides algorithms to find an optimal policy π^* with corresponding optimal state value function V^* as a perfect model of the MDP.

An optimal policy is the policy with the highest value function for all states of the environment. Q-Learning [7] is one of the most popular RL methods. It involves in finding state-action qualities rather than just state value. Assume that $Q^*(s_t, a_t)$ be the expected discounted reward of taking action a_t in state s_t, and then choosing actions optimally afterward. An estimation of optimal state-action value function, $Q^*(s_t, a_t)$, denoted by $\widetilde{Q}(s_t, a_t)$ and the Temporal-Difference (TD) is used to update it:

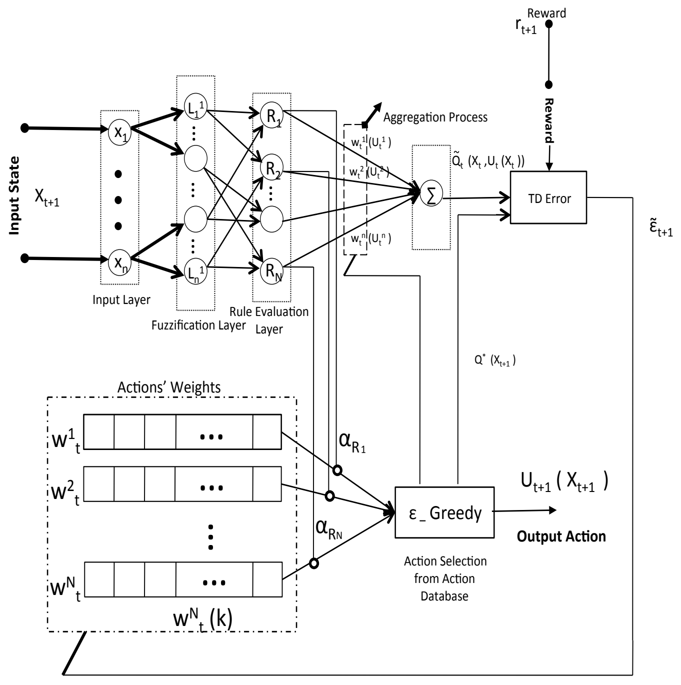


Fig. 1. FQL Structure.

$$\widetilde{Q}_{t+1}(s_t, a_t) = \widetilde{Q}_t(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a \widetilde{Q}_t(s_{t+1}, a) - \widetilde{Q}_t(s_t, a_t)]. \quad (1)$$

Here $0 < \alpha < 1$ is a learning rate.

Fuzzy Q-Learning (FQL) is the fuzzy extension of Q-Learning. It is capable of handling continuity in the state space. In addition, it is powerful enough to tackle the curse of dimensionality and other ordinary RL issues rising in real life and industrial problems. FQL, by the use of a Fuzzy Inference System (FIS), partitions each continuous state space variable and creates fuzzy rules. Each rule in the rule-base has a set of discrete actions associated with it. Each action in each rule has a quality factor (a weight), which will be adjusted throughout the learning phase. Also each rule has a weight vector based on its action qualities. Briefly speaking, FIS estimates the Q-value function for the current state-action pair. This Q-value function, along with the optimal Q-value of the state, is calculated in the same way. They will be used to compute the TD Error. Later on, based on both the TD Error and the update rule of TD learning, the action weights will be updated to gain more reinforcement. This procedure is similar to Q-Learning. The agent then chooses actions based on the quality values (weights) of the different actions available in the action set of each rule, along with an exploration/exploitation mechanism named double ε-Greedy. You can see the FQL structure in Figure 1 and find more details in [8] [9].

V. System Model

Our target system is an IaaS environment represented by a large scale data center including N heterogeneous physical nodes, and is based on the CloudSim architecture [10]. Each node is characterized by its CPU performance, disk storage, amount of RAM and network bandwidth. The software layer of the system is tiered, comprising local and global managers. Local managers reside on each physical node as a module of the VM monitor. They monitor the node CPU utilization, resize VMs according to their resource needs and decide when and which VMs have to be migrated from their host node. The global manager resides on a master node and gathers information from the local managers to manage the allocation of VMs by issuing VM migration commands and changing the power states of the nodes.

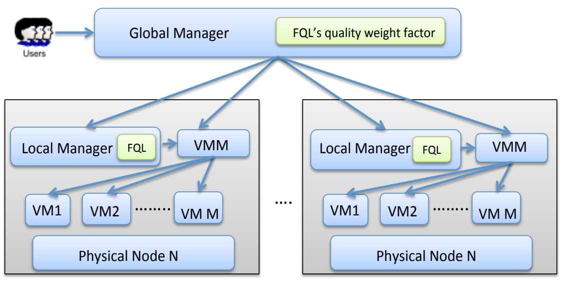


Fig. 2. System Model.

In our system model (see Figure 2), each physical node is associated with an FQL agent. The agent by interaction with dynamic characteristics of its corresponding host node learns, which VM selection criterion must be chosen for optimizing the energy-performance trade-off. Since our problem has a large input space, FQL is likely to suffer from a low learning convergence rate. In order to speed up convergence, we introduce cooperative leaning by sharing the FQL's quality weight factors among all physical hosts inside the data center.

VI. Formulating the FQL Task

Before formulating the criterion selection as an FQL task we address some metrics and definitions. To present a description of the energy-performance tradeoff, we must present a definition for the energy consumption and the cloud performance separately. In our system model the energy consumption (EC) of a server is defined as a linear function of the CPU utilization, and the cloud performance is defined as a function that evaluates the Service Level Agreement (SLA) delivered to any VM deployed in an IaaS. In our system model we employed two metrics to measure the SLA violations in a data center: SLA Violation due to Overutilization (SLAVO) and SLA violation due to Migrations (SLAVM) [2]. These metrics were defined with this assumption that the SLAs are delivered when 100% of the performance requested by applications inside a VM is provided at any time, bounded only by the parameters of the VM. In fact, SLAVO is the percentage of time, during which the active hosts have experienced a CPU utilization of 100% and SLAVM is the SLA violation by VMs due to live migration:

$$SLAVO = \frac{1}{N}\sum_{i=1}^{M}\frac{T_{s_i}}{T_{a_i}} \tag{2}$$

$$SLAVM = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{d_j}}{C_{r_j}} \tag{3}$$

Here N is the number of active hosts, T_{s_i} is the total time during which host i has experienced CPU utilization of 100%, T_{a_i} is the total time during which host i is serving VMs, M is the number of VMs, C_{d_j} is an estimate of the performance degradation of the VM j caused by migration (10% in our experiments), and C_{r_j} is the total CPU capacity requested by VM j during its life time. Therefore, a metric for describing SLA violations can be denoted by the product of SLAVO and SLAVM as follows:

$$SLAV = SLAVO \times SLAVM \tag{4}$$

This metric can encompass both performance degradation due to host overloading and VM migrations. Consequently, to represent the energy-performance trade-off we use the product combination of SLA Violation and Energy consumption (EC) that the authors of [2] denoted as the total Energy SLA violation (ESV):

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

Considering the aforementioned equations we can define SLAVM and SLAVO as well as ESV for each host as follows:

$$SLAVO_i = \frac{T_{s_i}}{T_{a_i}}, \qquad 1 \leq i \leq N \tag{6}$$

$$SLAVM_i = \frac{1}{K_i}\sum_{j=1}^{K_i}\frac{C_{d_{ji}}}{C_{r_{ji}}}, \qquad 1 \leq i \leq N \tag{7}$$

$$SLAV_i = SLAVO_i \times SLAVM_i, \qquad 1 \leq i \leq N \tag{8}$$

$$ESV_i = SLAV_i \times EC_i, \qquad 1 \leq i \leq N \tag{9}$$

Here K_i is the number of VMs that reside on host i, $C_{d_{ji}}$ is an estimate of the performance degradation of the VM j caused by migration from host i (10% in our experiments), and $C_{r_{ji}}$ is the total CPU capacity requested by VM j during its life time inside host i. The goal of the integration of multiple VM selection criteria is to optimize the energy-performance tradeoff. Therefore, the FQL task is formulated as follows:

The Reward Function. The optimal long-term cumulative reward is the target of the FQL. The reward is a performance feedback on the resulted new VM selection criterion chosen from the action set. Therefore, the reward function can be defined as follows:

$$Reward_{i_{t+1}} = \frac{1}{ESV_{i_{t+1}}}, \qquad 1 \leq i \leq N \tag{10}$$

The Input State. The FQL must be able to trace the physical host's behavior by merely observing the input states. Consequently, dynamic characteristics of the host must be included into the input state, which have enough information about the host's behavior. In addition they must be able to make inter-dependency between state and actions. In this paper the host's resources and the number of VMs residing on the host were considered as the element of the input state. These are time-varying factors caused by dynamic workloads and live migration. Therefore, the input state is defined as follows:

$$X_{t_i} = \{CU_{t_i}, NumVM_{t_i}\}, \qquad 1 \leq i \leq N \tag{11}$$

Here, CU_{t_i} is the CPU Utilization of the host i in time-step t, and $NumVM_{t_i}$ is the number of VMs residing on the host i in time-step t.

The Action. Each element of the action set denotes a VM selection criterion. Therefore, it can be defined as follows:

$$A\big(X_t\big) = \{Criterion_1, Criterion_2, ..., Criterion_n\} \tag{12}$$

When the VM selection task is called for a host, the FQL agent associated with this host observes the current state S_{t_i} and gathers information about the input state X_{t_i} (including the current CPU utilization and the number of VMs) and immediately choses a VM selection criterion as an action from the action set $A\big(X_t\big)$ in order to make a decision. One time-step later, the FQL agent receives a numerical reward, $Reward_{i_{t+1}}$ and finds itself in a new state S_{t+1}. In a trial and error interaction, finally the FQL agent learns how to map the states to the actions in order to maximize the discounted sum of rewards. In other words, the FQL agent learns the best sequence of VM selection criteria, corresponding to the states observed during its lifetime.

VII. Experimental Setup

In our experiments the CloudSim toolkit has been chosen as simulation platform. Real life data on workload are provided by the CoMon project, which monitors the infrastructure of PlanetLab [11]. We simulated a data center comprising 800 heterogeneous physical nodes, half of which are HP ProLiant ML110 G4, and the other half consisting of HP ProLiant ML110 G5 servers. The power consumption of the selected servers is different at each load level. The frequencies of the servers' CPUs are mapped onto MIPS rating. Each server is modeled to have 1 GB/s network bandwidth and the characteristics of the VMs types corresponding to Amazon's EC2 instance types including High-CPU Medium Instance, Extra Large Instance, Small Instance and Micro Instance. The Initialization of VMs is done according to the resource requirements defined by the VM types. We used three different workloads, which were collected in a timespan of three different days. During the simulation each VM is randomly assigned a workload trace from one of the VMs from the corresponding day. In our dynamic VM consolidation procedure, for all of the experiments we used a static threshold based algorithm [2] to detect host overloading detection, Power Aware Best Fit Decreasing (PABFD) algorithm [2] for the VM placement and for host underloading detection we used a simple strategy that selects the host with the minimum utilization compared to the other hosts. In the FQL setup, we used three fuzzy sets in the FIS configuration with Gaussian membership functions for each input set element. The elements of the action set are:

$$A(X_t) = \{MINU, MAXU\} \tag{13}$$

MAXU (Maximum Utilization) means that the VM selection task selects VMs having the highest utilization. *MINU (Minimum Utilization)* means that the VM selection task selects VMs, having the lowest utilization. The learning rate of Q-Leaning has been experimentally set to 0.1, and the quality factors of actions (weights) were initialized randomly. The FQL-iteration or time-step for perceiving next state and receiving the reward of the previous state has been set to 300 seconds.

VIII. Experimental Results

In the first experiment we compare the results of the dynamic VM consolidation procedure when its VM selection task uses the FQL strategy with the same procedure when it uses a random strategy. Both strategies integrate MAXU and MINU criteria together with one fundamental difference, the FQL strategy benefits from a learning procedure to choose the criteria intelligently but the random strategy employs them randomly without any knowledge. In this comparison we have measured the total ESV value per iteration for the random strategy and compared it to the FQL strategy. This experiment has been evaluated with the same workload for both procedures. In addition, the result of the procedure with the random strategy is an average of five independent simulations. Since host overloading and underloading, as well as VM placement policies are the same in both procedures, the curves depicted in Figure 3 show us the effect of the two different strategies described above. As Figure 3 shows, the FQL strategy in the preliminary iterations acted without any learning knowledge. Therefore, it is natural that its result is very close to the random strategy or even worse than it. But after a while, the FQL strategy following a trial and error interaction learns how to find a policy (mapping states to actions), to maximize the discounted sum of rewards obtained. Indeed, this policy is a sequence of criteria where decision making for selecting VM based on them leads to the maximization of the reciprocal of the Energy SLA Violation (ESV) value discussed in Section VI.

In the next experiment we repeat the previous experiment, with this difference that we compare the result of the FQL strategy with fixed criterions (either MINU or MAXU) for making decisions (see Figure 4). It shows how FQL learns the appropriate policy to choose VM selection criteria, to utilize the synergetic contributions of them to effectively decrease the ESV value during the simulation.

In the third experiment we compared all metrics involved to represent our energy-performance tradeoff when the VM selection task uses the FQL strategy with ones uses minimum utilization and maximum utilization as a fixed criterion. This experiment was evaluated with three different workloads on

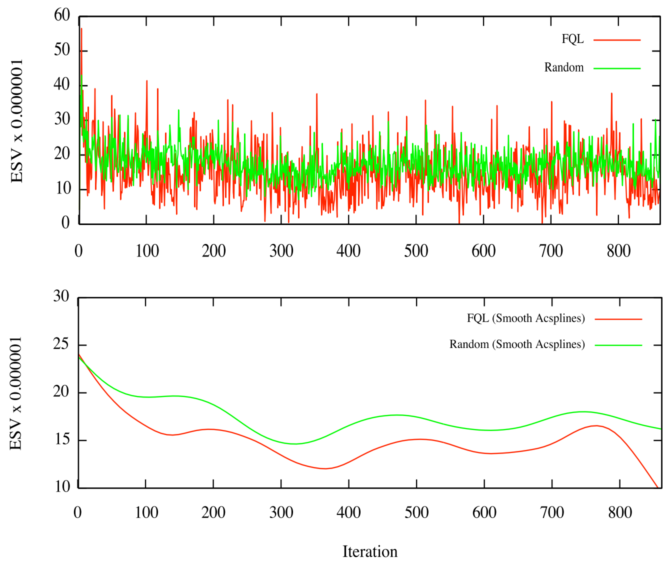


Fig. 3. Comparison between the FQL strategy and the random strategy

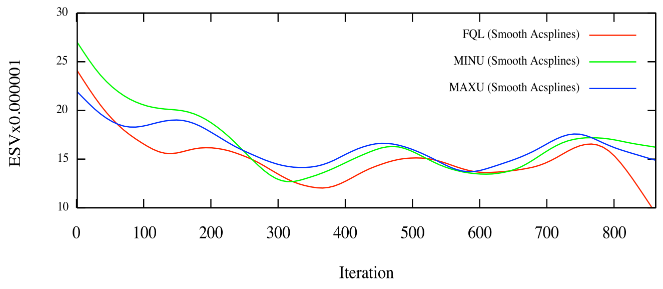


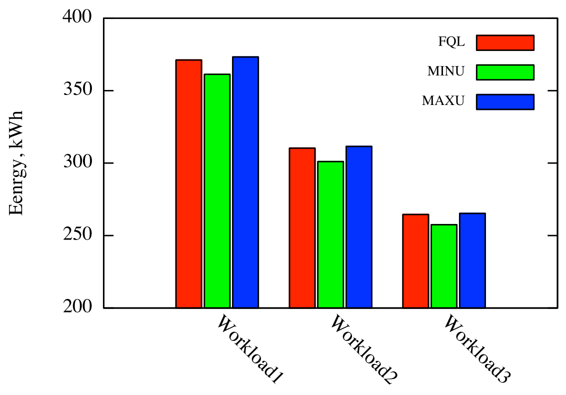
Fig. 4. Comparison between VM selection using the FQL strategy and VM selection using fixed criteria

three different days. Figures 5 (a) and (b) show the total value of the energy consumption and the total value of SLA violations respectively, Figure 5 (c) illustrates ESV as a tread-off value between energy and SLA violations. These figures show that the FQL strategy has learned how to find a sequence of applying MAXU and MINU criteria to select VMs during dynamic VM consolidation procedure in order to keep balance between energy consumption and SLA violation and thus improves the ESV.

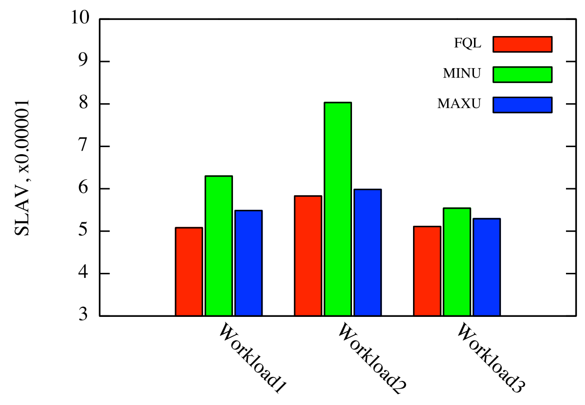
Figure 6 shows the SLA violation metrics (SLAVM and SLAVO) and the number of migrations during simulation. It is clear to see that how the FQL applies different criteria to control the number of migrations and SLA violations.

IX. Conclusion

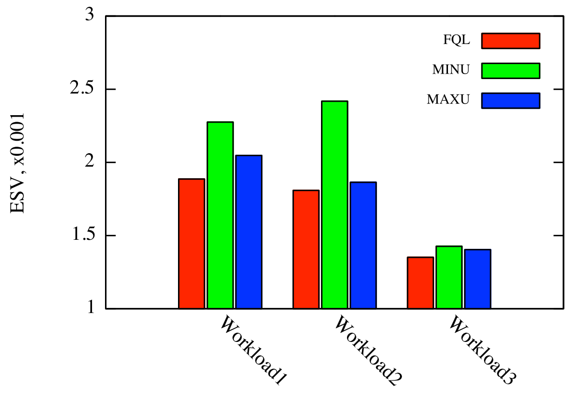
In this paper we propose a Fuzzy Q-Learning (FQL) approach as an online decision making strategy to enhance the VM selection task in a dynamic VM consolidation procedure. Our approach is able to integrate multiple VM selection criteria to benefit from all advantages and possible synergetic contributions of them in long-term learning. In fact, our approach learns how to find an optimal strategy to apply-



(a)

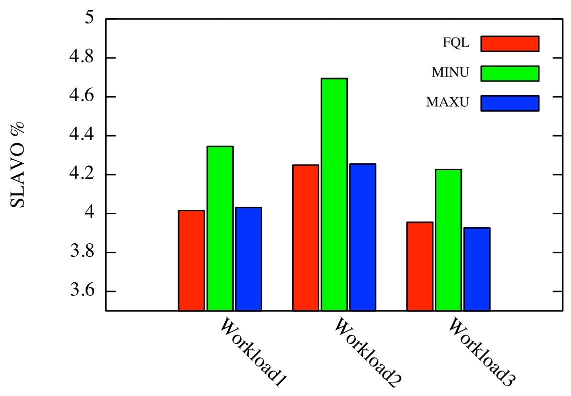


(b)

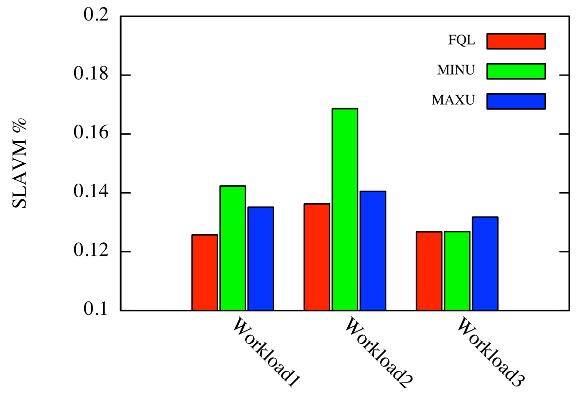


(c)

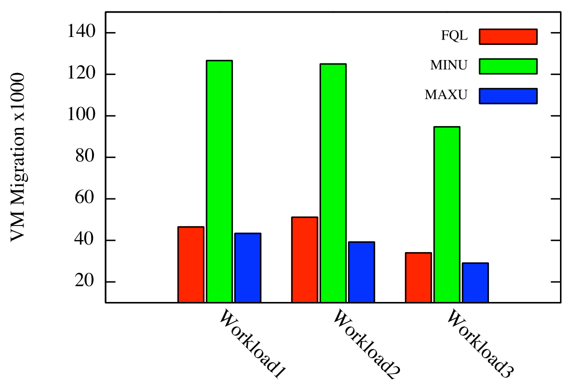
Fig. 5. Energy SLA violations



(a)



(b)



(c)

Fig. 6. SLA violation metrics and number of migrations

ing multiple criteria for selecting VMs during a dynamic VM consolidation procedure towards improving the energy-performance trade-off. As our system model has a multi-agent system architecture, we employed a cooperative learning strategy in order to speed up learning convergence rates and consequently better decision making. Our results illustrate that our proposed approach outperforms VM selection policies using fixed criteria for decision making. As a part of our future work, we aim at implementing this idea using more VM selection criteria, which can for example be based on bandwidth and memory utilization, etc. In addition we would like to implement this idea in a real-life cloud environment with an open source cloud management framework such as OpenStack Neat [12] and Snooze [13].

References

[1] The Climate Group, "SMART 2020: Enabling the Low Carbon Economy in the Information Age," in *Report on behalf of the Global eSustainability Initiative*, 2008.

[2] 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.

[3] ——, “Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers,” in *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*. ACM, 2010, p. 4.

[4] ——, “Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints,” *Parallel and Distributed Systems, IEEE Transactions on*, no. 99, pp. 1–14, 2012.

[5] G. Khanna, K. Beaty, G. Kar, and A. Kochut, “Application performance management in virtualized server environments,” in *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*. IEEE, 2006, pp. 373–381.

[6] A. Beloglazov, J. Abawajy, and R. Buyya, “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.

[7] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[8] L. Jouffe, “Fuzzy inference system learning by reinforcement methods,” *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 28, no. 3, pp. 338–355, 1998.

[9] A. F. Naeeni, “Advanced Multi-Agent Fuzzy Reinforcement Learning,” Degree Project, Dalarna University, 2004.

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

[11] 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.

[12] A. Beloglazov and R. Buyya, “Openstack neat: A framework for dynamic consolidation of virtual machines in openstack clouds–a blueprint,” Technical Report CLOUDS-TR-2012-4, Cloud Computing and Distributed Systems Laboratory, The University of Melbourne, Tech. Rep., 2012.

[13] E. Feller, L. Rilling, and C. Morin, “Snooze: A scalable and autonomic virtual machine management framework for private clouds,” in *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*. IEEE Computer Society, 2012, pp. 482–489.