2015 International Conference on Cloud and Autonomic Computing

Self-Learning Cloud Controllers: *Fuzzy Q-Learning for Knowledge Evolution*

Pooyan Jamshidi*, Amir M. Sharifloo†, Claus Pahl*, Andreas Metzger†, Giovani Estrada ‡
*Department of Computing, Imperial College London, UK
†University of Duisburg-Essen, Germany
‡Intel, Ireland

***Abstract*—Auto-scaling features enable cloud applications to maintain enough resources to satisfy demand spikes, reduce costs and keep performance in check. Most auto-scaling strategies rely on a predefined set of rules to scale up/down the required resources depending on the application usage. Those rules are however difficult to devise and generalize, and users are often left alone tuning auto-scale parameters of essentially black-box applications. In this paper, we propose a novel fuzzy reinforcement learning controller, FQL4KE, which automatically scales up or down resources to meet performance requirements. The Q-Learning technique, a model-free reinforcement learning strategy, frees users of most tuning parameters. FQL4KE has been successfully applied and we therefore think that a fuzzy controller with Q-Learning is indeed a promising combination for auto-scaling resources.**

I. Introduction

Elasticity is a key characteristic of cloud platforms enabling both cloud providers and end-users to set policies to satisfy demand spikes, minimize costs, and increase platform utilization and application performance [14], [19]. Despite the advantages, however, the dynamic acquisition and release of resources remains a challenge due to the inherent uncertainty introduced by workloads, costs and user constraints (e.g., response time).

A large number of approaches have been proposed with varying degree of success [19], [20], [8]. Scaling rules based on predefined thresholds are often simple and intuitive to set up, see for instance Amazon EC2 [1], Microsoft Azure [2] and OpenStack [3]. This thresholding mechanism, one size fits all approach, however, makes it difficult to generalize and quickly adapt to different scenarios. As the applications grow in complexity, the interference among components and the frequency by which hardware and software failures arise impose new challenges [19], [21], [11]. In the remaining sections we introduce a novel fuzzy controller which together with Q-Learning automatically updates fuzzy rules to learn optimal elasticity policies at runtime.

We have previously shown that a fuzzy auto-scaling controller can successfully enhance the user experience by allowing intuitive auto-scaling decisions [15]. The key strength of fuzzy logic is their ability to translate human knowledge into a set of basic and understandable rules. During the design process of a fuzzy controller, a set of `IF-THEN` rules must be defined. These rules represent the mapping of the input received from monitoring to the output scaling decisions entangled with the actuator in linguistic terms. Although users are more comfortable with defining fuzzy auto-scaling rules using linguistic variables [15], the rules have to be defined at design-time based on limited knowledge available. Here is where learning strategies could be employed to ease the burden at design time.

In this paper we propose a fuzzy online learning mechanism, `FQL4KE` that adjusts auto-scaling policies at runtime. More specifically, we combine fuzzy control and Fuzzy Q-Learning (FQL) [13] in order to connect human expertise to continuous evolution machinery. The main implication of this work is that users need not rely on unreliable guesswork, but rather `FQL4KE` automatically adjusts application resources, with no apriori knowledge on the auto-scaling actions. It mean the auto-scaling controller can indeed start working with an *empty* knowledge base and adjusting it on the fly.

The rest of the paper is organized as follows. Section II presents our approach. Section III gives the experimental results. Section IV discusses the related work, and finally Section V concludes the paper.

II. The Proposed Approach

This section presents our approach `FQL4KE` to cloud auto-scaling with machine learning [1]. By combining fuzzy logic and Q-Learning, our approach deals with uncertainty caused by the incomplete *knowledge* of cloud users. Expert knowledge, if available, is encoded in terms of fuzzy rules. The fuzzy rules are continually tuned through learning from the data collected by monitoring runtime executions. In case there is no knowledge available at design time to provide the initial fuzzy rules, `FQL4KE` is still able to learn the rules at runtime.

A. FQL4KE Architecture

Figure 1 illustrates the main building blocks of `FQL4KE`. While the application runs on a cloud platform that provides the demanded resource, `FQL4KE` monitors the application and guides resource provisioning. More precisely, `FQL4KE` follows the autonomic MAPE-K loop [17], where different characteristics of the application (e.g. workload and response time) are continuously monitored, the satisfaction of system goals are checked and accordingly the resource allocation is adapted in case of deviation from goals. The goals (i.e., SLA, cost, response time) are reflected in a reward function.

[1] An ellaboated description of `FQL4KE` is available in [16].

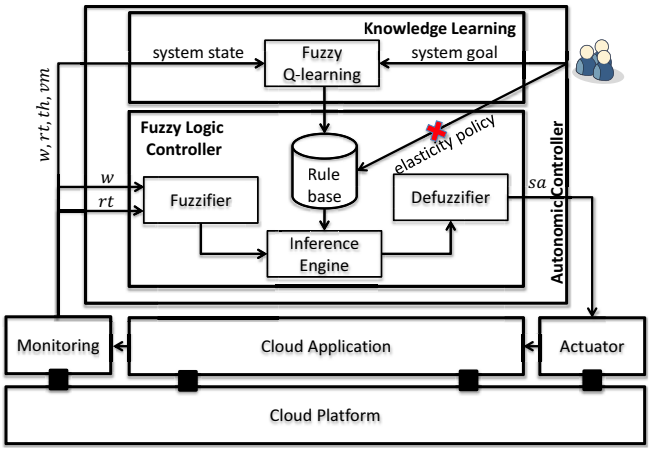

Fig. 1: FQL4KE architecture.

The monitoring component collects low-level performance metrics and feed both cloud controller as well as the knowledge learning component. The actuator issues adaptation commands that it receives from the controller at each control interval to the underlying cloud platform. Two components of knowledge learning and cloud controller are incorporated for this purpose. The cloud controller is a fuzzy controller that takes the observed data, and generates scaling actions. The learning component continuously updates the knowledge base of the controller by learning appropriate rules. These two components are described in Sections II-B and II-C respectively. Finally, the integration of these two components is discussed in Section II-D.

B. Fuzzy Logic Controller

Fuzzy inference is the process of mapping a set of control inputs to a set of control outputs through fuzzy logic rules and operations. This mapping provides a basis from which control output can be derived. The key benefit of fuzzy controllers is for types of problems that cannot be represented by explicit mathematical models due to high non-linearity of the system. Instead, the potential of fuzzy logic lies in its capability to approximate that non-linearity by expressing the knowledge in a similar way to the human perception and reasoning.

The explicit knowledge base component is one of the unique aspects of such type of controllers. Instead of sharp switching between modes based on thresholds, control output changes smoothly from different region of behavior depending on the dominant rule [15]. Since our goal is to build a cloud controller to adjust the number of computing machines with regard to the response time and workload, our fuzzy controller is designed based on a model that a set of input signals are mapped on an output control action. More precisely, the input to the controller is the current workload (w) and response time (rt) and the output is the scaling action (sa) in terms of increment (or decrement) in the number of virtual machines (VMs).

C. Fuzzy Q-Learning

Fuzzy Q-Learning component is to learn/adjust/adapt the auto scaling policies used by the controller. The implication is that we do not anymore rely on user-defined policies, instead we let the controller learn the policies. As the controller has to take an action in each control loop, it should try to select those actions taken in the past which produced good rewards.

Fuzzy logic version of Q-learning algorithm has been proposed first in [13] with the aim to optimize the consequents of the rules in fuzzy controllers. Fuzzy Q-learning (FQL) has some critical benefits over its traditional algorithm. First and most importantly, for some application areas in which the number of states and the potential action that the agent can take in those states are high then the q-values [2] need to be stored in large look up tables. As a result, the Q-learning becomes unpractical and even impossible to solve in continuous state spaces [13]. By employing fuzzy variables, continuous state spaces can be discretized into states represented by all the combinations of inputs. In addition, FQL can be speed up by embedding prior knowledge via fuzzy rules.

D. FQL4KE for Dynamic Resource Allocation

`FQL4KE` starts with controlling the allocation of resources with no priori knowledge. After enough *explorations*, the consequents of the fuzzy rules can be determined by selecting those actions that corresponds to the *highest* q-value in each row of the Q table. Although `FQL4KE` do not rely on design-time knowledge, if even partial knowledge is available (i.e., operator of the system is confident with providing some of the elasticity policies) or there exists data regarding performance of the application, `FQL4KE` can exploit such knowledge by initialing q-values with more meaningful data instead of initializing them with zero. This implies a faster learning convergence.

III. Experiment

`FQL4KE` has been implemented on top of Microsoft Azure, as a promising commercial Cloud platform. The cloud controller deployed in a delayed-feedback environment came to know the reward after a non-negative integer indicating the number of time-steps between taking an scaling action and actually receiving its feedback (the state observation and reward). In each monitoring cycle, which happens every 10 seconds, the controller knows about its state but in order to receive the reinforcement signal, it has to wait for example for 8-9 minutes for ”scaling out” actions and 2-3 minutes for ”scaling in” actions to be enacted. These numbers vary depending on the Cloud platform. Such kind of delayed feedback environments introduce some challenges for learning convergence. Therefore, finding an optimal balance between exploration and explotation is vital. Following an intuitive strategy, FQL starts with exploration phase and after a first learning convergence happened, it enters the balanced exploration-exploitation phase. In other words, after initial

[2]Q-values estimate the award of taking an action in the long run.

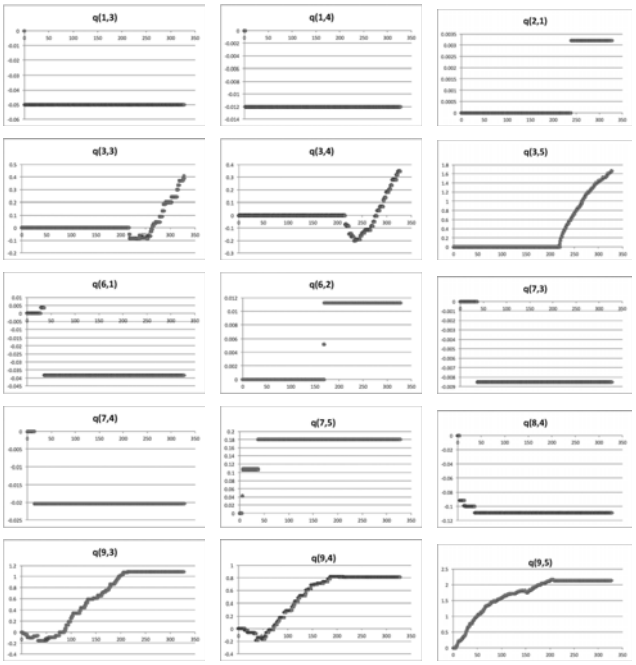

Fig. 2: Temporal evolution of q-values.

learning by high exploration, we increase the exploitation rate to fully exploit the learned knowledge.

In practice, a finite rounds of learning steps (in machine learning terminology called *epochs*) should be passed to ensure the learning has been converged. In our experiment, the exploration lasts 80 epochs, ensuring that all state-action pairs are sufficiently visited. The temporary evolution of the q-values associated to each state-action pairs for the learning strategy is shown (for partial set of pairs) in Figure 2. Note that the change in the q-values occurs when the corresponding rule is activated, i.e., when the system is in state $S(t)$ and take specific action a_i. As the figure shows, some q-values changed to a negative value during exploration phase. It means that these actions are basically punished and as a result are not appropriate to be taken in the future. The optimal consequent for each rule in the rule base is determined by the most highest q-value at the end of the learning phase. For instance, action a_5 is the best consequent for rule number 9 in learning strategy.

In accordance to the change in the q-values, the control surface of the fuzzy controller is also evolving. Figure 3 shows the temporal evolution in control surface of the fuzzy controller. The initial design-time surface is not shown as it is a constant plane at point zero. The surface is evolved until the learning has been converged. Note that the first surface is the one in the lower left, then lower right, upper right and the final surface is located at the upper left corner when the learning has been converged.

IV. Related Work

Auto scaling problem has been extensively studied in the recent years [19][7][10][12][4]. However, prior to this work

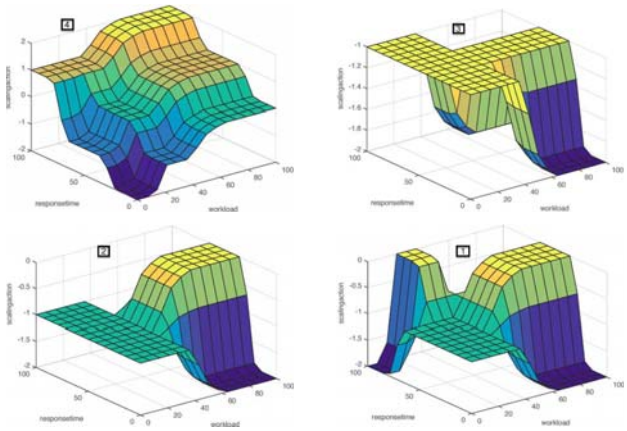

Fig. 3: Temporal evolution of control surface.

there has been no technique to tune and improve auto scaling policies. In this section, we overview the works, which apply kind of learning techniques in the area of resource allocation, and distinguish our approach from them.

Xu et al. [24], [6] present an approach to learning appropriate auto-configuration in virtualized resources. It uses multiple agents, each of which apply reinforcement learning to optimize auto-configuration of its dedicated environment. Barrett et al. [5] investigate the impact of varying performance of cloud resources on application performance. They show that a resource allocation approach, considering this aspect, achieves benefits in terms of performance and cost. To reduce the learning time, a parallelized reinforcement learning algorithm is proposed through which multiple agents are employed to deal with the same tasks to speed up the procedure to explore the state space. The reward values are calculated by combining the accumulated experience of different agents. In a similar approach [9] appropriate initialization of the q-values are proposed to speedup the learning convergence. Tesauro et al. [23] demonstrate how to combine the strengths of both RL (model-free) and queuing models (model-based) in a hybrid approach, in which their RL needs to be trained at design time while at runtime a queuing model policy controls the system.

In [22], a multi-layer approach is presented to handle multi-objective requirements such as performance and power in dynamic resource allocation. The lower layer focuses on each objective, and exploits a fuzzy controller proposed earlier in [25]. The higher layer is to maintain a trade-off by coordinating the controllers. Lama et al. [18] integrate Neural Networks (NNs) with fuzzy logic to build adaptive controllers for autonomic server provisioning. Similar to our approach, NNs define a set of fuzzy rules, and the self-adaptive controller adapts the structure of the NN at runtime, therefore automatically updating rules. Unlike the above approaches, `FQL4KE` offers a seamless knowledge evolution through fuzzy control and RL, putting aside the burden that was on the shoulder of users.

V. Conclusions and Future Work

This paper presents a novel knowledge evolution strategy for the dynamic resource provisioning of cloud-based applications. The scenario under study assumes no a priori knowledge regarding elasticity policies. More precisely, instead of specifying elasticity policies as a typical case-by-case scenario in auto-scaling solutions, system operators and users are only required to provide the importance weights in the reward functions. A fuzzy rule-based controller together with a Q-Learning algorithm will then iterate to learn optimal elasticity policies at run-time. The main contributions of the proposed approach are as follows:

1) `FQL4KE` is *robust* to highly dynamic workload intensity due to its self-adaptive and self-learning capabilities.
2) `FQL4KE` is *model-independent*. The variations in the performance of the deployed applications and the unpredictability of dynamic workloads do not affect the effectiveness of the proposed approach.
3) `FQL4KE` is capable of automatically constructing the control rules and keeping control parameters updated through *fast online learning*. It executes resource allocation and learns to improve its performance simultaneously.
4) Unlike supervised techniques that learn from the training data, `FQL4KE` *does not require off-line training* that saves significant amount of time and efforts.

We are currently extending our prototype in a number of ways, for instance: (i) extending `FQL4KE` to perform in the environments where only partially observable (for this we will exploit partially observable Markov decision processes) data exist; (ii) exploiting clustering approaches to learn the membership functions of the antecedents in fuzzy rules; and (iii) integrating `FQL4KE` with advanced data sensors to profit from the underlying hardware platform and ensure it runs exceptionally well on Intel architecture. The association of low-level features and resource allocation, eg seen through the scheduler, can drive the knowledge evolution towards optimal policies that factor in hardware features and co-processors. Elastic policies could then be reused by the service orchestrator on similar workloads (eg via scheduler hints) to speed up the overall learning convergence.

Overall, initial experiments show that our approach can be successfully applied to cloud auto-scaling. We therefore expect our work can bring attention to this concept in which adaptive, self-tuning applications automatically optimize the resource allocation in cloud environments.

Acknowledgment

The project has been funded by the Irish Centre for Cloud Computing and Commerce, as well as the European Commission's 7th Framework Programme projects CloudWave (agreement 610802) and LeanBigData (agreement 619606).

References

[1] *Amazon EC2*, Accessed December 29, 2014.
[2] *Microsoft Azure*, Accessed December 29, 2014.
[3] *OpenStack*, Accessed December 29, 2014.
[4] Danilo Ardagna, Giuliano Casale, Michele Ciavotta, Juan F Pérez, and Weikun Wang. Quality-of-service in cloud computing: modeling techniques and their applications. *Journal of Internet Services and Applications*, 5(1):11, 2014.
[5] Enda Barrett, Enda Howley, and Jim Duggan. Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. *Concurrency and Computation: Practice and Experience*, 25(12):1656–1674, 2013.
[6] Xiangping Bu, Jia Rao, and Cheng-Zhong Xu. Coordinated self-configuration of virtual machines and appliances using a model-free learning approach. *IEEE Trans. Parallel Distrib. Syst.*, 24(4):681–690, 2013.
[7] E Caron, L Rodero-Merino, F Desprez, and A Muresan. Auto-scaling, load balancing and monitoring in commercial and open-source clouds. 2012.
[8] Eddy Caron, Luis Rodero-Merino, Frédéric Desprez, Adrian Muresan, et al. Auto-scaling, load balancing and monitoring in commercial and open-source clouds. In *Cloud Computing Methodology, Systems, and Applications*, 2012.
[9] X Dutreilh and S Kirgizov. Using reinforcement learning for autonomic resource allocation in clouds: towards a fully automated workflow. *The Seventh International Conference on Autonomic and Autonomous Systems*, 2011.
[10] Guilherme Galante and Luis Carlos E. de Bona. A Survey on Cloud Computing Elasticity. In *2012 IEEE Fifth International Conference on Utility and Cloud Computing*, pages 263–270. IEEE, November 2012.
[11] A Gambi, M Pezze, and G Toffetti. Kriging-based Self-adaptive Cloud Controllers. *IEEE Transactions on Services Computing*.
[12] Alessio Gambi, Giovanni Toffetti, and Mauro Pezzè. Assurance of self-adaptive controllers for the cloud. In *Assurances for Self-Adaptive Systems*, pages 311–339. Springer, 2013.
[13] PY Glorennec and L Jouffe. Fuzzy Q-learning. In *Proceedings of 6th International Fuzzy Systems Conference*, volume 2, pages 659–662. IEEE, 1997.
[14] Pooyan Jamshidi, Aakash Ahmad, and Claus Pahl. Cloud Migration Research: A Systematic Review. *IEEE Transactions on Cloud Computing*, 1(2):142–157, 2013.
[15] Pooyan Jamshidi, Aakash Ahmad, and Claus Pahl. Autonomic resource provisioning for cloud-based software. In *SEAMS*, pages 95–104, 2014.
[16] Pooyan Jamshidi, Amir Sharifloo, Claus Pahl, Andreas Metzger, and Giovani Estrada. Self-learning cloud controllers: Fuzzy q-learning for knowledge evolution. *arXiv preprint arXiv:1507.00567*, 2015.
[17] JO Kephart and DM Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, January 2003.
[18] P Lama and X Zhou. Autonomic provisioning with self-adaptive neural fuzzy control for percentile-based delay guarantee. *ACM Transactions on Autonomous and Adaptive Systems*, 2013.
[19] T Lorido-Botran, J Miguel-Alonso, and JA Lozano. A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments. *Journal of Grid Computing*, 2014.
[20] Marco AS Netto, Carlos Cardonha, Renato LF Cunha, and Marcos D Assunçao. Evaluating auto-scaling strategies for cloud computing environments. *Target*, 80:60.
[21] Xing Pu, Ling Liu, Yiduo Mei, Sankaran Sivathanu, Younggyun Koh, Calton Pu, and Yuanda Cao. Who Is Your Neighbor: Net I/O Performance Interference in Virtualized Clouds. *IEEE Transactions on Services Computing*, 6(3):314–329, July 2013.
[22] Jia Rao, Yudi Wei, Jiayu Gong, and Cheng-Zhong Xu. Dynaqos: Model-free self-tuning fuzzy control of virtualized resources for qos provisioning. In *Quality of Service (IWQoS), 2011 IEEE 19th International Workshop on*, pages 1–9, June 2011.
[23] G Tesauro, NK Jong, R. Das, and M.N. Bennani. A Hybrid Reinforcement Learning Approach to Autonomic Resource Allocation. *2006 IEEE International Conference on Autonomic Computing*, 2006.
[24] Cheng-Zhong Xu, Jia Rao, and Xiangping Bu. URL: A unified reinforcement learning approach for autonomic cloud management. *J. Parallel Distrib. Comput.*, 72(2):95–105, 2012.
[25] Jing Xu, Ming Zhao, J. Fortes, R. Carpenter, and M. Yousif. On the use of fuzzy modeling in virtualized data center management. In *Autonomic Computing, 2007. ICAC '07. Fourth International Conference on*, pages 25–25, June 2007.