Horizontal and Vertical Self-Adaptive Cloud Controller with Reward Optimization for Resource Allocation

Jesus Alejandro Cardenes[1], Doina Precup[2] and Ricardo Sanz[3]

CAR

(1) Centre for Automation and Robotics (CAR), Universidad Politécnica de Madrid (UPM). (2) Reasoning and Learning Lab, McGill University. (3) Autonomous Systems Laboratory, UPM.

Abstract

Over-booking or under-booking of computing resources leads to higher cost and performance degradation of web applications. To optimize the performance of web applications, access to the resources has to be dynamically controlled ensuring maximum cost-performance ratio of the application while fulfilling requirements. To simplify the design of dynamic cloud controllers, we propose an horizontal and vertical scalability self-aware agent defined by a self-adaptive fuzzy logic with an oriented random optimizer based on reward and memory. The algorithm dynamically adjusts the membership functions and their relationship, maximizing the reward of the system while considering the cost related to the deployment of new resources. The evaluation of the controller under real cloud workload reveals the ability of the algorithm to maximize the performance of the web application based on the target parameters given by an operator.

Objectives

The following research presents and highlights the usage of an oriented random optimizer based on reward and memory, ORORM, for the development of a self-adaptive fuzzy logic , FL, horizontal and vertical cloud controller. The optimizer dynamically updates the parameters defining the linguistic terms of a FL. The main novelties presented are the abilities of the ORORM to:

- Dynamically fine-tune the cloud controller without human intervention or pre-knowledge of the current or future behaviour of the web application controlled. Ensuring the expected quality of service of the web application.
- Minimize the instabilities derived from the simultaneous vertical and horizontal elasticity adjustment

Self-adaptive Cloud Controller

The presented ORORM for the FL allows the system to quickly respond to sudden changes on the workload affecting the web application while experimenting with the relation between the parameters of each membership function. This behaviour is achieved thanks to the quick respond obtained from the orientation of the search of the parameters of the linguistic terms of the FL and the posterior fine-tuning executed by the random search based on the reward. Also the use of memory allows the controller to increase the importance of recent high rewarded actions compare to old actions.

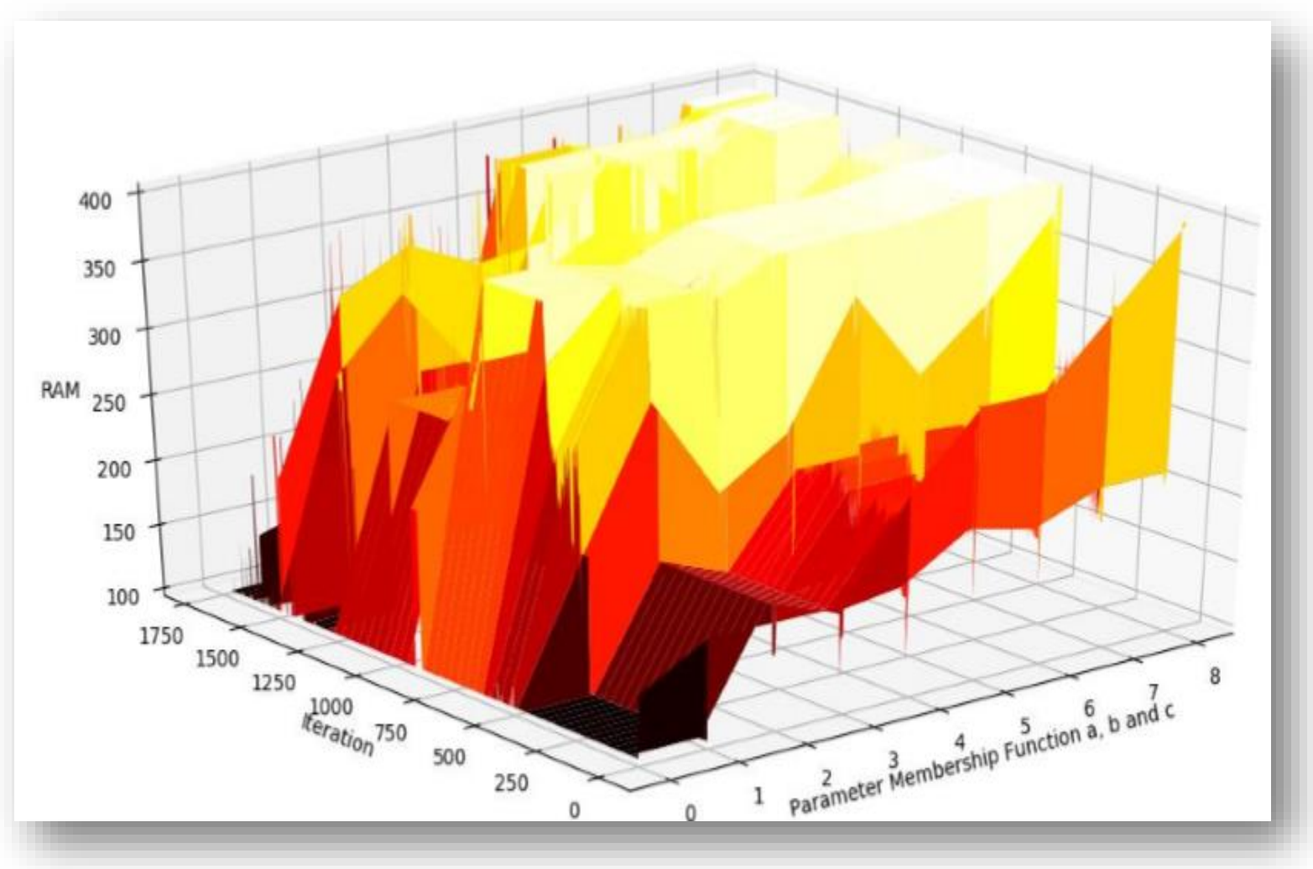

Figure 1. Fuzzy sets learning. It represents the evolution of each one of the parameters of the membership functions and the relationship between them.

Experiments

To evaluate the performance of the oriented random optimizer for the development of a self-adaptive cloud controller, we have tested the behavior of the RUBIs [1] web application under 32 hours of the workload from Wikipedia (extension '.de') [2]. The algorithms evaluated have been:

- Baseline fuzzy logic cloud controller.
- Self-adaptive FL, SAFL: Baseline fuzzy logic with ORORM.
- Greedy self-adaptive FL: Exploits high rewarded actions.

The main parameters evaluated during the simulation have been the percentage of the memory used, the memory assigned per container, the number of containers deployed (1 or 2) and the number of HTTP connections during the simulations.

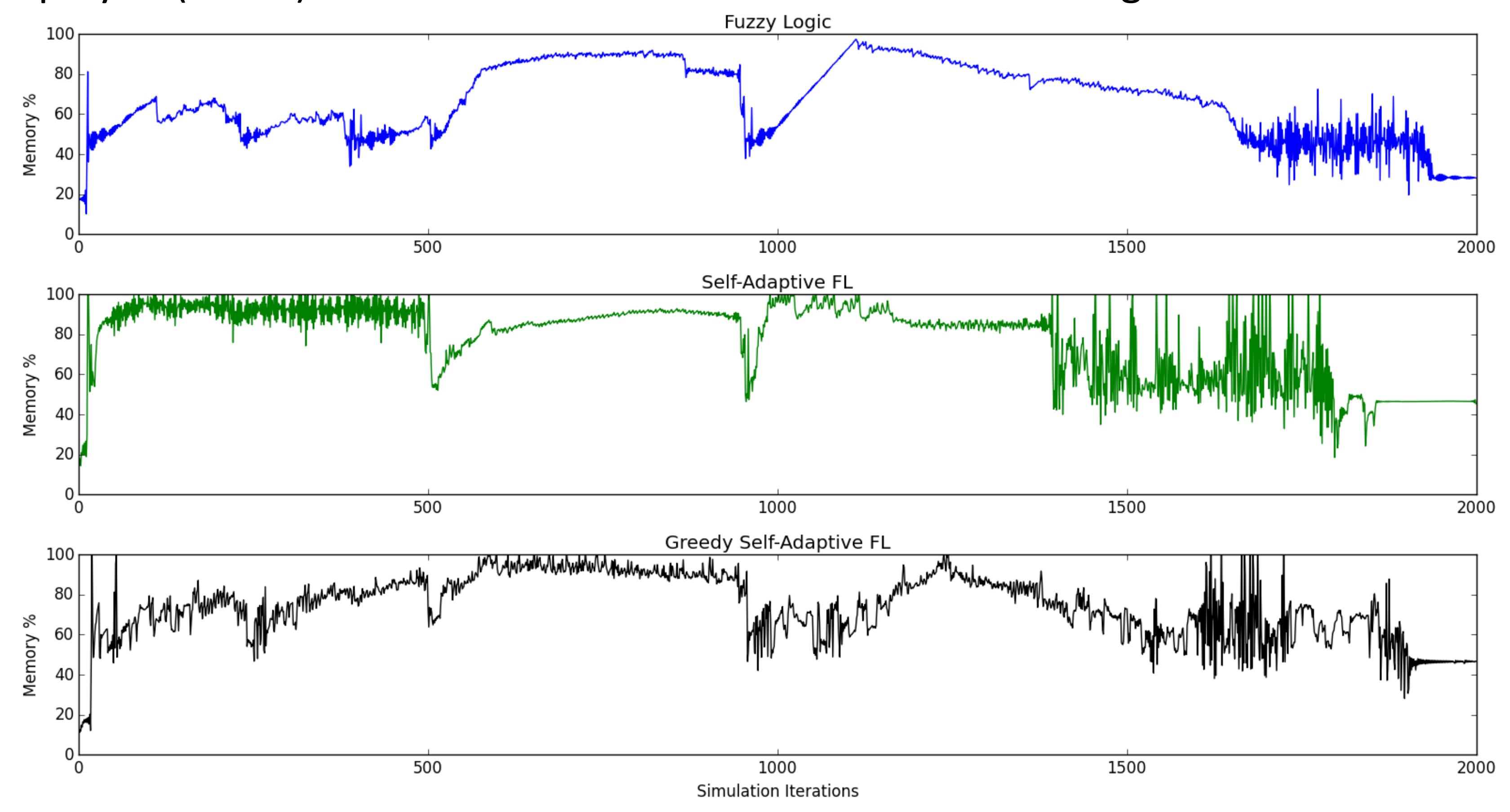

Figure 2. Percentage of Memory used per Cloud Controller during simulation.

At the top part of figure 1, it shows the baseline percentage of the memory used by the web application under the usage of a fuzzy logic controller. At the middle and bottom part of the figure, it is possible to see the enhancement of the percentage of memory used during the full simulation thanks to the optimization done by the Self-adaptive and Greedy Self-adaptive FL. The SAFL is able to maximize the percentage of memory used since the beginning of the simulation while the Greedy SAFL is able to decrease the fluctuations during the full simulation.

Table I

Memory %	Mean	Median	Std. Memory %
FL	68.02	67.59	16.80
Self-adaptive FL, SAFL	78.68	85.74	18.32
Greedy SAFL	77.15	78.74	13.59

Table I presents the percentage of the memory used by the web application under each cloud controller simulated. This table shows the ability of the self-adaptive fuzzy logic, SAFL, to learn the space and improve the percentage of the memory used. Nevertheless, the SAFL generated the highest number of oscillations, as verified thanks to the calculus of the standard deviation. On the other hand, the greedy SAFL not only improves the memory usage but also has the lowest number of memory usage oscillations during the simulation.

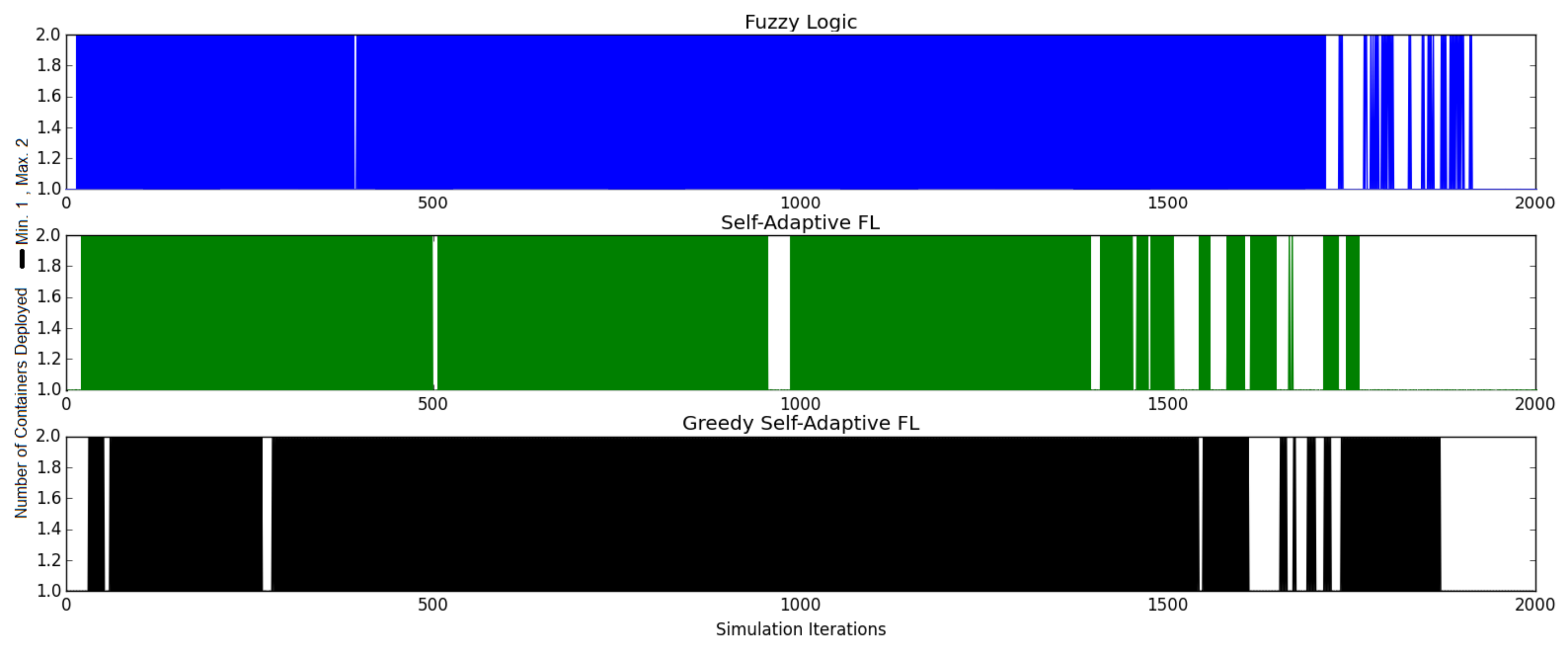

Figure 3. Nº containers used per Cloud Controller during simulation. Colors blue, green and black indicate 2 containers deployed while color white, 1 container.

Table II and figure 3 evaluates the benefits of the ORORM to decrease the instabilities related to fast and inconsistent deployment of containers. Thanks to the evaluation of the time maximizing the usage of one container and the amount of time that one or two containers are deployed is possible to measure the performance of the three cloud controllers.

Table II

Containers Iterations maximizing the use of a unique container, IMUUC	Percentage IMUUC, full simulation	Time on state with one container deployed, Mean	Time on state with one container deployed, Median FL
146	7,92%	8,81	6
299	17,41%	16,18	12
203	9,86%	14,375	11,5

Conclusion and Related Work

Previous research done by Soodeh et al. [3] presented a FL controller able to coordinate the right amount of CPU and memory. However as presented in their paper, it was needed to obtain feedback from experts and then empirically tune the rules of the FL. Therefore, the main advantage of the proposed ORORM compared to previous research is the ability of the system to dynamically self-adapt the algorithm and improve the performance of the web application.

Regarding horizontal elasticity, Ahmad Al-Shishtawy et al. [4] presented the combination of a feedforward and a feedback controller. A feedback controller responsible of responding to sudden changes on the workload while in parallel a feedback algorithm fine tunes the response of the controller. The main advantage of our presented method is that by combining two feedback loops, one for the controller and another for the optimizer there is no need to create a high level logic combining two independent and complex controllers as presented in [4].

References

[1] RUBIS: Rice University Bidding System. Available at http://rubis.ow2.org/index.html.
[2] Urdaneta et al. "Wikipedia Workload Analysis for Decentralized Hosting," Volume: 53, Number 11, July 2009, Elsevier Computer Networks.
[3] Soodeh Farokhi et al. "Coordinating CPU and Memory Elasticity Controllers to Meet Service Response Time Constraints," Vienna University of Technology, Austria, ICCAC 2015.
[4] Ahmad Al-Shishtawy et al. "ElastMan: Autonomic Elasticity Manager for Cloud-Based Key-Value Stores," in HPDC '13 Proceedings of the 22nd international symposium on High-performance parallel and distributed computing.

The presented development and cloud control sandbox used for this research can be found at:

https://github.com/Chuseuiti/CloudControlSandbox