2016 IEEE International Conference on Autonomic Computing

Self-Adaptive Capacity Controller: A Reinforcement Learning Approach

Luis Tomás
Department of Computing Science
Umeå University, Sweden
Email: luis@cs.umu.se

Seyed Saeid Masoumzadeh, Helmut Hlavacs
Faculty of Computer Science
University of Vienna, Austria
Email: {seyed.saeid.masoumzadeh, helmut.hlavacs}@univie.ac.at

***Abstract*—Interference between co-located VMs may lead to performance fluctuations and degradation. To limit this problem, VMs access to physical resources needs to be controlled to ensure certain degree of isolation among them. This mapping between virtual and physical resources must be performed in a dynamic way so that it can be adaptive to the changing applications requirements, as well as to the different set of co-located VMs. To address this problem we propose a self-adaptive fuzzy Q-learning capacity controller that proactively readjusts the isolation degree based on applications performance. Our evaluation demonstrates a reduction into VMs interference and an increment on the overall utilization, while still ensuring critical applications performance, and providing more resources to non-critical applications.**

I. Introduction and Background

When servers utilization increase, the chances of VMs interfering with each other also increase, problem known as *noisy neighbor problem* [1]. Applications' tolerance to interference are different [2], being deadline constrained applications not much affected by transient resource shortages, but being interactive applications strongly affected if they need to maintain a certain minimum performance all the time. Hence making difficult to assess the possible impact of overload situations.

Existing solutions try to avoid/reduce their impact, either through migrations or by reducing the overbooking pressure upon applications performance fluctuations [2] [3]. However, most of them lower overall utilization while the main problem is VM interference and not too high utilization. Other solutions present an interference-aware scheduling [1], however that may not be adaptive enough for quickly changing workloads or VMs very sensitive to interference.

To overcome these difficulties, different isolation levels need to be created at a server-level, meaning that some VMs will have access to some physical resources in isolation while others may share these resources with other VMs – thus providing QoS differentiation. This can be achieved by limiting VMs access to resources through KVM core pinning [4]. However, current mechanisms lack the flexibility and knowledge to adapt the isolation levels based on application requirements over time. Hence they cannot properly handle the trade-off between ensuring application performance and keeping the server utilization rate high.

In this work we propose a self-adaptive performance-aware capacity controller, able to adapt the isolation levels between critical and non-critical VMs by dynamically changing the mapping between virtual cpus and physical cpus based on applications performance. We exploited a fuzzy reinforcement learning algorithm, known as Fuzzy Q-Learning (FQL) [5], which allows us to deal with the high complexity of cloud systems, as well as with all their uncertainties, specially regarding future applications needs. Our experimental results highlight the more efficient resource usage made by our FQL controller, which improves the capacity sharing decisions, leading to a rise in the utilization ratio without neither hurting critical nor non-critical applications performance.

II. Fuzzy Q-Learning Capacity Controller

An initial QoS differentiation schema was presented in [6], where different QoS levels were created for different applications by pinning or not VMs to a set of exclusive or shared physical cores. To vary the isolation degree and help to increase utilization, the exclusive access to the set of physical cores is progressively reduced until the application needs them again – when the complete isolation is recovered. This is a conservative, reactive approach that may impact both critical VMs, as well as non-critical VMs due to the abrupt (and frequent) reductions in the amount of cores that are shared among VMs. To address this problem we propose a self-adaptive capacity controller that proactively adapts the isolation levels based on an online reinforcement learning technique called Fuzzy Q-Learning (FQL) [5] – combination of Q-Learning (QL) and fuzzy logic.

A. *Fuzzy Q-Learning Model*

Reinforcement learning (RL) [7] comprises a set of machine learning methods designed to address a particular learning task in which an agent is placed in an unknown environment and is allowed to take actions/make decisions which can change its state in the environment and bring it delayed numerical reward. The goal of the agent is to learn a policy that tells it what action/decision to take/make in order to maximize the cumulative reward over time. Q-learning (QL) [7] is a popular RL algorithm that represents the learning knowledge by means of a Q-table, whose Q-values are defined for each state-action pair, reflecting the expected cumulative reward of taking that action in that state and following the optimal policy thereafter. Q-table can be updated and learned by *temporal-difference* (TD) [7] method. Therefore, an optimal policy can be constructed by simply selecting the action with the highest value in each state. For our approach, we utilize Fuzzy Q-Learning (FQL) [5] as a fuzzy extension of QL that helps to tackle "curse of dimensionality", also allowing us to encapsulate expert knowledge into the learning table resulting in speeding up the learning process.

B. *FQL-based Capacity Controller*

To take advantage of the FQL, our capacity controller applies the isolation actions (i.e., core repining actions) based on the FQL agents decisions. There is an FQL agent associated to each critical VM, making dynamic decisions about their resource isolation needs locally, even though they have no information about the status of the resources at the scale of the physical host or other co-located VMs. To do so, a combination of the current response time rt_t and the current number of isolated cores nc_t describe each state inside critical VMs. Each element of the action set denotes the number of cores needed in isolation, for transiting from current state to the next state. As FQL's goal is to optimize the trade-off between sharing capacity and performance over time, the reward function returns a value determining a success rate, here maximizing **the number of shared cores** and minimizing **the response time**:

$$r_{t+1} = (1 - nc_t/tc) - (rt_t/tr), \tag{1}$$

where tc is the total number of physical cores assigned to a critical VM, and tr is the target response time.

1) Let's reactive and proactive be friends: Owing to the learning capacity, the FQL agents can predict applications behavior and increase/decrease their exclusive access to resources before the performance degradation happen. As Q-Learning (QL) agents learn by trial and error interaction with the environment, it is likely to degrade the performance due to either random actions in an exploration mode or lacking of knowledge in its Q-table during the learning process. Our proposed controller can guarantee the performance for the critical VMs after the learning process but during the learning or under completely new situations there is no guarantee. Although, combining the Q-learning with the fuzzy logic can speedup the learning process, such performance degradation may not be acceptable for critical VMs, even for a short-term. To address this problem, our proposed learning controller calls a reactive controller to act on its behalf once the response time exceeds the set target threshold. In this situation the reactive controller establishes the maximum isolation level for the VM again, similarly to [6]. In addition, the learning controller receives a punishment proportional with the amount of exceedance (see Eq. (1) where $rt > tr$) once it calls the reactive controller. Therefore, the learning controller also learns how to act to decrease the number of calls over time.

III. Performance Evaluation

The performance of the capacity allocation controller is evaluated in a testbed, where a representative cloud workload is emulated by mixing different types of VMs, grouped in critical and non-critical applications. We use *RUBiS*[1] and *RUBBoS*[2] cloud benchmarks to emulate the critical (interactive) VMs. They are big long living VMs, with seasonality patterns in their use – following real traces, in our case Wikipedia traces[3]. For the non-critical applications we generated different type of relatively short living, non-interactive VMs. We execute different set of VMs that continuously solve random sudokus[4], together with a mix of VMs which run different shell scripts that consume random amount of CPU and memory over time, increasing the uncertainty in the system and thus generating a more realistic workload. As regard to the FQL tunning, we have empirically set configuration variables to: (a) equally consider short- and long-term rewards; (b) dynamic adapt the learning rate to apply punishment faster than reward; (c) and take decisions mostly based on already gathered knowledge but forcing learning actions every now and then.

To evaluate the performance of the FQL capacity management controller we measured the response time of the critical VMs, the throughput achieved by the non-critical ones, the number of concurrently running VMs, and the overall server utilization. We have compared it with the controller implemented in [6], which already outperformed the standard KVM scheduler by better isolating VMs. The results obtained highlight the superior performance of the FQL controller, which achieves an overall reduction on the response times for critical VMs, specially during workload peaks, as well as reduces performance fluctuations (both in number and size) – leading to a more stable and predictable system. Additionally, the differences grow bigger and bigger over time thanks to the FQL acquired learning. The FQL agents are able to reduce critical VMs isolation when their performance is not affected, even without knowledge about the status of the infrastructure, and therefore achieving a more efficient resource sharing. Thanks to that, more physical resources can be used by the non critical VMs, leading to more concurrently running applications (5% more in an already overbooked system), consequently increasing the overall utilization – without hurting non-critical VMs performance either.

IV. Conclusions

Existing isolation techniques, such as KVM core pinning, are commonly used to alleviate the impact VMs may have on other co-located VMs. However, they lack the knowledge and/or the adaptivity needed to handle the trade-off between not impacting VMs performance and keeping servers utilization high. To tackle this problem, in this paper we present a Fuzzy Q-Learning-based capacity controller that dynamically decides how much capacity each critical VM can share with non-critical VMs, increasing overall utilization while maintaining VMs performance.

Acknowledgements

This collaborative research was initiated with an STSM granted by IC1304 COST-ACROSS action. This work was partially supported by the Swedish Research Council (VR) for the project Cloud Control.

References

[1] L. Tomás, C. Vázquez, J. Tordsson, and G. Moreno, "Reducing noisy-neighbor impact with a fuzzy affinity-aware scheduler," in *ICCAC*, 2015.

[2] L. Tomás, C. Klein, J. Tordsson, and F. Hernández-Rodríguez, "The straw that broke the camel's back: safe cloud overbooking with application brownout," in *ICCAC*, 2014.

[3] R. Householder, S. Arnold, and R. Green, "On cloud-based oversubscription," *Intl. Journal of Engering Trends and Technology (IJETT)*, vol. 8, no. 8, pp. 425–431, 2014.

[4] IBM Knowledge Centre, "Kernel Virtual Machine (KVM): Best practices for KVM," Web page at http://www-01.ibm.com/support/knowledgecenter/linuxonibm/liaat/liaatbestpractices_pdf.pdf?lang=en.

[5] P. Y. Glorennec and L. Jouffe, "Fuzzy q-learning," in *Intl. Conference on Fuzzy Systems*, vol. 2, 1997, pp. 659–662.

[6] L. Tomás and J. Tordsson, "Cloud Service Differentiation in Overbooked Data Centers," in *UCC*, 2014.

[7] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 1998.

[1] http://rubis.ow2.org
[2] http://jmob.ow2.org/rubbos.html
[3] http://dumps.wikimedia.org/other/pagecounts-raw/
[4] http://norvig.com/sudoku.html