Rethinking Reinforcement Learning for Cloud Elasticity*

Konstantinos Lolos, Ioannis Konstantinou, Verena Kantere and Nectarios Koziris
School of Electrical and Computer Engineering, National Technical University of Athens
{klolos,ikons,nkoziris}@cslab.ece.ntua.gr,verena@dblab.ece.ntua.gr

ABSTRACT

Cloud elasticity, i.e., the dynamic allocation of resources to applications to meet fluctuating workload demands, has been one of the greatest challenges in cloud computing. Approaches based on reinforcement learning have been proposed but they require a large number of states in order to model complex application behavior. In this work we propose a novel reinforcement learning approach that employs adaptive state space partitioning. The idea is to start from one state that represents the entire environment and partition this into finer-grained states adaptively to the observed workload and system behavior following a decision-tree approach. We explore novel statistical criteria and strategies that decide both the correct parameters and the appropriate time to perform the partitioning.

1 APPROACH

Modern large-scale computing environments based on either virtualized or bare-metal resources like private cloud clusters, public cloud deployments and data centers deploy tenths of complex platforms like NoSQL and database servers, web farms, etc. on thousands of machines and execute hundreds of services [5].

A crucial issue in such environments both in terms of cost and performance is the correct allocation of resources to platforms and applications so that they are neither over-provisioned, nor under-provisioned, aiming to avoid both resource saturation and idling, and having as utmost goal fast execution of user workload while keeping the cost of operating the infrastructure as low as possible.

Managing the above trade-off is challenging. First, the number of system and application parameters (such as number and type of computing nodes, configuration settings such as replication factor, workload metrics such as observed throughput/latency, etc.) that affect performance is exceedingly large; therefore, the number of possible states of the system, which correspond to combinations of different values for all such parameters is exponentially large.

Second, parameter values may have an unknown discrete or continuous value range (e.g., cluster and load characteristics, live performance metrics, etc.), requiring methods to automatically perform a correct discretization that optimally captures system behavior using a small state number. Third, many times it is not known beforehand if and how a parameter or parameter value affects system behavior according to user defined metrics. Fourth, the time interval between two consecutive resource management decisions is usually at least in the order of minutes, reducing the collection rate of training data. Nevertheless, the resource management technique should be able to work with little such data. All four challenges are hard to address even for static workloads and applications, and become insurmountable for dynamic ones.

Public cloud providers such as Amazon, Google, Microsoft and IBM offer autoscaling services [1]. These employ threshold-based rules to regulate infrastructural resources (e.g., if mean CPU usage is above 40% then add a new VM). However, such solutions do not address any of the aforementioned challenges. Some research approaches also explore threshold-based solutions [4]. More sophisticated approaches employ *Reinforcement Learning* (RL) algorithms such as *Markov Decision Processes* (MDP) and *Q-Learning* which are natural solutions for decision making problems and offer optimality guarantees under conditions, yet they suffer from limitations that derive from the assumption of a priori parameter knowledge and their role to system behavior, as well as from the curse of dimensionality as a result of their effort to create a full static model of the computing environment [3].

In this work we employ RL in a novel manner that starts from one global state that represents the environment, and gradually partitions this into finer-grained states adaptively to the workload and the system behavior; this results in a state space that has coarse states for combinations of parameter values (or value ranges) for which the system has unchanged behavior and finer states for combinations for which the system has different behavior. Our technique can zoom into regions of the state space where the system changes behavior, and use this information for elasticity decisions.

We propose a novel decision tree-based algorithm[1], that dynamically partitions the state space when needed. This allows the algorithm to work on a multi-dimensional continuous state space, but also to adjust the state space size based on the amount of information on the system behavior. The algorithm takes as input criteria for splitting the states, which aim to partition the existing behavior information, with respect to the measured parameter values, into subsets that represent the same behavior. We propose two novel criteria, the *Parameter test* and the *Q-value*. The algorithm adopts strategies that perform splitting of one or multiple states, employing small or big amounts of information on behavior. More information can be found here [2].

REFERENCES

[1] AWS | Auto Scaling. https://aws.amazon.com/autoscaling/.
[2] Lolos, K., et al. Elastic Resource Management with Adaptive State Space Partitioning of Markov Decision Processes. *arXiv:1702.02978 [cs]* (Feb. 2017).
[3] Rao, J., et al. VCONF: a Reinforcement Learning Approach to Virtual Machines Auto-configuration. In *ICAC* (2009), ACM, pp. 137–146.
[4] Shen, Z., Subbiah, S., Gu, X., and Wilkes, J. Cloudscale: Elastic Resource Scaling for Multi-Tenant Cloud Systems. In *SoCC* (2011), ACM, p. 5.
[5] Verma, A., et al. Large-scale Cluster Management at Google with Borg. In *EuroSys* (2015), ACM, p. 18.

*This paper is supported by European Union's Horizon 2020 RIA programme under GA No 690588, project SELIS.

SoCC '17, September 24–27, 2017, Santa Clara, CA, USA

ACM ISBN 978-1-4503-5028-0/17/09.
https://doi.org/10.1145/3127479.3131211

[1] Available at https://github.com/klolos/reinforcement_learning