# Learn-as-you-go with Megh: Efficient Live Migration of Virtual Machines

Debabrota Basu*, Xiayang Wang†, Yang Hong†, Haibo Chen†, Stéphane Bressan*
*School of Computing, National University of Singapore, Singapore
†Institute of Parallel and Distributed Systems, Shanghai Jiao Tong University, Shanghai, China

***Abstract*—We propose a reinforcement learning algorithm, Megh, for live migration of virtual machines that simultaneously reduces the cost of energy consumption and enhances the performance. Megh learns the uncertain dynamics of workloads as-it-goes. Megh uses a dimensionality reduction scheme to project the combinatorially explosive state-action space to a polynomial dimensional space. These schemes enable Megh to be scalable and to work in real-time. We experimentally validate that Megh is more cost-effective and time-efficient than the MadVM and MMT algorithms.**

## I. Introduction

Infrastructure as a Service (IaaS) environments of Cloud computing allocate Virtual Machine instances (VM) on a cluster of physical machines to provide a shared platform of resources. The uncertain dynamics of workloads, creating abrupt and unpredictable changes in resource utilization, requires dynamic allocation of VMs. In order to avoid such disruption of service, [1] proposed the idea of a live migration scheme. Each migration decision depends on three questions: *when* to move a virtual machine, *which* virtual machine to move and *where* to move it? Dynamic, uncertain and heterogeneous workloads running on virtual machines and large-scale of data centers make such decisions difficult. Knowledge-based and heuristics-based algorithms are commonly used to tackle this problem. Knowledge-based algorithms, such as MaxWeight scheduling algorithms, are dependent on the specifics and the dynamics of the targeted Cloud architectures and applications. Heuristics-based algorithms, such as MMT algorithms [2], suffer from high variance and poor convergence because of their greedy approach. We propose a reinforcement learning [3] approach. This approach does not require prior knowledge. It learns the dynamics of the workload as-it-goes. We formulate the problem of energy- and performance-efficient resource management during live migration as a Markov decision process. While several learning algorithms are proposed to solve this problem, these algorithms remain confined to the academic realm as they face the curse of dimensionality. They are either not scalable in real-time, as it is the case of MadVM [4], or need an elaborate offline training, as it is the case of Q-learning. We propose, Megh to overcome these deficiencies. Megh uses a novel dimensionality reduction scheme to project the combinatorially explosive state-action space to a polynomial dimensional space with a sparse basis. Megh has the capacity to learn uncertain dynamics in real-time. Amalgamation of the projection scheme with learning enables Megh to be both scalable and robust. A detailed description of the methodology and the experiments is available in [5].

## II. Live migration as a decision problem

A Cloud data center consists of a number of physical machines. They host a certain number of VMs depending on their specifications and capacities. Each VM allocates certain physical resources according to the requirement of each user. The energy consumed by each of the physical machines is proportional to the total CPU performance, memory and disk storage used by the VMs allocated to it. We utilize the SPECpower benchmark [2] to model the energy consumption. Live migration aids to dynamically allocate and update the resources to VMs. But live migration may cause service disruption by incurring downtime and migration time. Through the service level agreements the Cloud provider legally promises not to cross, a predefined threshold of downtime and migration time. The performance is modelled by imposing penalties if the service level agreements are violated.

Leveraging this cost structrure, we model efficient live migration as a *Markov decision process* [3]. In this model, states are the configurations of VMs operating on a set of physical machines and their corresponding workloads. Since the workloads vary continuously and are uncertain, the state space becomes infinite dimensional and introduces uncertainty in state transitions. An action is defined as migration of a VM from one physical machine to the other. The *cost* of evolving from one configuration to the other by migrating a VM is given by sum of the costs of energy consumption and SLA violation in this time interval. In order to quantify the uncertainty, we need to define a transition function that returns the probability to reach a state from a given state if a certain migration is performed. But in our problem, the transition function is not known *a priori* and has to be learned. If the transition function is known a priori, then the future effect of an action is expressed using the expectation of its discounted cumulative cost. The effect of an action on the future configurations decays by a discount factor with each time step. Thus, if we have a policy that determines what action to take from a given configuration, the cost-to-go function of following that policy from an initial state. The *cost-to-go function* is given by the expected sum of the discounted costs of the migrations from the beginning to an infinite horizon. Thus, the problem manifests in finding the *optimal policy* that minimizes the cost-to-go function.

## III. Megh and learning as-you-go

The intuitive approach to find the optimal policy is to start with an initial policy and to update it gradually using dynamic programming. This scheme is called policy iteration [3]. But it

Table I
PERFORMANCE EVALUATION FOR PLANETLAB

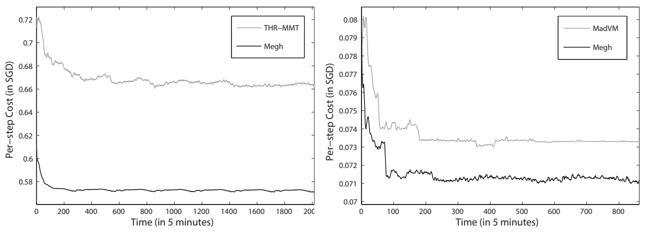
| Algorithms | THR-MMT | IQR-MMT | MAD-MMT | LR-MMT | LRR-MMT | Megh |
|---|---|---|---|---|---|---|
| Total cost (SGD) | 1347 | 1504 | 1367 | 1392 | 1392 | 1155 |
| #VM migrations | 325299 | 444624 | 331304 | 324079 | 324079 | 2309 |
| #Active physical machines | 666 | 684 | 682 | 692 | 692 | 203 |
| Execution time (ms) | 2016 | 3077 | 2226 | 1924 | 2080 | 1426 |

is not practical to search through the whole state-action space that consists of all possible configurations of VMs on all the physical machines. This *curse of dimensionality* of state-action space makes the computation expensive and almost infeasible to perform in real-time. The expectation for cost-to-go function is also infeasible to calculate because the stochastic nature of workloads, their correlation with VM configurations and their transitions are not known *a priori*. Since restricting this workload dynamics to a specific transition function breaks the robustness and universality of learning approach, we avoid it.

In order to tackle the curse of dimensionality, we project the state-action space to a space of, the number of physical machines times the number of VMs, dimensions. Each basis vector of this space corresponds to the migration of a VM to a specific physical machine. Thus, all the configuration changes can be represented using these basis vectors or combinations of them. The basic rationale behind this projection is that while transitioning from one state to the other, we can reach only the states that are one action away from the present state. Thus rather than searching over the whole state space in each step, it is sufficient to search among the states reachable from the present one. Now, we approximate the cost-to-go function to a linear projection on this space. Still the expectation of the cost-to-go function is not computable due to lack of any prior knowledge of the transition function. Thus, we construct a *transition operator* that accumulates the possibility of to migrate from the present configuration to the other depending on the nature of workload and the corresponding changes. We begin with a transition operator that allows the system to migrate any of the VMs to any of the machines without bias. As the system extract information of the workload and VM configurations with time, it decides an action according to the policy. Thus, we update the operator to incorporate the effect of present state and action and its discounted influence in future action. Megh plugs in these two schemes to Least-Square Policy Iteration algorithm [6]. Thus, Megh first constructs a projection-based estimation of cost-to-go function by least-square estimation and then updates the policy such that it maximizes the estimate.

## IV. PERFORMANCE EVALUATION

We validate effectiveness and efficiency of Megh using the CloudSim toolkit. CloudSim contains workloads extracted from the CoMoN project which was monitoring the geo-distributed computing platform, PlanetLab [7]. We use the standard price of Singapore EMA, $0.18675$ SGD/kWh, to calculate the energy consumption cost. We assume that Cloud providers would pay back $16.7\%$ and $33.3\%$ of user's money depending on whether the performance degradation is less than or greater than $0.10\%$. Though it is a bit costlier than reality, it does not harm the analysis. Table I depicts that Megh outperforms the MMT algorithms on a week-long trace of PlanetLab. Since MadVM fails to scale-up for the complete PlanetLab in our experimental facilities, we choose two random sets of 150 workloads running on 100 physical machines for 3 days of PlanetLab trace. Figure 1(a) shows Megh reaches the almost stable cost per-step in around $100$ steps because of the initial learning phase. Being a greedy heuristics, THR-MMT faces high variance and instability even after the initial convergence. Figure 1(b) shows MadVM and Megh, being reinforcement learning algorithms, have similar trends of convergence. But Megh reduces the expenditure by $14.25\%$ and $4.3\%$ with respect to that of THR-MMT and MadVM respectively. It validates the cost-effectiveness, robustness and stability of Megh for optimal resource management for a diverse set of workloads with respect to other heuristics. Megh also speeds up $1.41$ times with respect to THR-MMT. Megh executes an iteration in 7ms whereas MadVM does the same in approximately $4143$ms. Four second is almost as same as the migration time of a VM of $0.5$GB RAM in the PlanetLab set-up. The curse of dimensionality and bookkeeping of a huge transition matrix make reinforcement learning algorithms, like MadVM, too slow to work in real-time. This empirically proves the efficiency of Megh not only as an effective learning algorithm but also as an eligible real-time resource management system in Clouds.



(a) THR-MMT vs. Megh (b) MadVM vs. Megh

Figure 1. Comparative per-step cost (in SGD) performance analysis.

## ACKNOWLEDGEMENT

This research project is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

## REFERENCES

[1] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the NSDI-Volume 2*, 2005, pp. 273–286.

[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," *Concurr. Comput. : Pract. Exper.*, vol. 24, no. 13, pp. 1397–1420, Sep. 2012.

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

[4] Z. Han, H. Tan, G. Chen, R. Wang, Y. Chen, and F. C. M. Lau, "Dynamic virtual machine management via approximate markov decision process," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, April 2016, pp. 1–9.

[5] D. Basu, X. Wang, Y. Hong, H. Chen, and S. Bressan, "Learn-as-you-go with megh: Efficient live migration of virtual machines," School of Computing, NUS, Tech. Rep. TRA4/17, April 2017.

[6] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *The Journal of Machine Learning Research*, vol. 4, pp. 1107–1149, 2003.

[7] 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.