ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/303519345

# Reinforcement Learning based Autonomic Virtual Machine Management in Clouds

**Conference Paper** · May 2016

DOI: 10.1109/ICIEV.2016.7760166

CITATION
1

READS
122

**2 authors:**

Arafat Habib
Chosun University
**10** PUBLICATIONS **3** CITATIONS
SEE PROFILE

Muhidul Islam Khan
Tallinn University of Technology
**17** PUBLICATIONS **66** CITATIONS
SEE PROFILE

**Some of the authors of this publication are also working on these related projects:**


Reinforcement Learning based Autonomic Route Optimaization in complex multi stage Supply Chain network View project


eLINK project funded by European Union View project

All content following this page was uploaded by Muhidul Islam Khan on 22 November 2017.

The user has requested enhancement of the downloaded file.

# Reinforcement Learning based Autonomic Virtual Machine Management in Clouds

Arafat Habib
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
e-mail: akhtab007@gmail.com

Muhidul Islam Khan
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
e-mail: muhit.islam.khan@gmail.com

***Abstract*— Cloud computing is a rapidly emerging field, services and applications are more or less 24/7. Resource dimensioning in this field is a great issue. Research is already going on to imply reinforcement learning to automate decision making process in case of addition, reduction, migration and maintenance of the Virtual Machines (VM) to balance the service level performance and VM management cost. Models have been proposed in this case based on Q-learning, a very popular reinforcement learning technique that is used to find optimal action selection policy for any finite Markov Decision Process (MDP). In this paper we propose to work with the challenges like proper initialization of the early stages, designing the states, actions, transitions using Markov Decision Process (MDP) and solving the MDP with two popular reinforcement learning techniques, Q-learning and SARSA(λ).**

***Keywords—cloud computing; markov decision process; reinforcement learning; virtual machines***

## I. INTRODUCTION

Cloud computing is one kind of computing that provides sharing functionalities/computing resources rather than having dedicated servers. Resource dimensioning in this field is a great issue. There are some research works to imply reinforcement learning to automate decision making process in case of addition, reduction, migration and maintenance of the Virtual Machines (VM) to balance the service level performance and VM management cost. For giving an on demand network access to a shared pool of computing resources, cloud computing is a great emerging model where resources are configurable on different parameters. Resources include networks, servers, applications, storage, etc. There are three sorts of service models of cloud computing. They are namely IaaS, PaaS and SaaS. IaaS provides the fundamental building blocks of computing resources. SaaS is the top layer of cloud computing services. It is typically built on top of a platform as a service solution and provides a software solution to the end users. Operating at the layer above raw computing hardware, whether physical or virtual, PaaS provides a method for programming languages to interact with services like databases, web servers and file storage [1]. IaaS takes the traditional physical computer hardware: such as servers, storage arrays and networking. It lets anyone build virtual infrastructure that mimics these resources but which can be configured, created, resized and removed within moments as a task requires it or the user wishes it [2]. Auto Scaling is another core feature of cloud computing that focuses on, on demand pulling and releasing of shared pool of available resources. It has a control loop monitor to decide if the system should grow or shrink. Our work mainly focuses on Virtual Machine management problem that can be used in IaaS and PaaS service models so that the system can perform auto scaling [3].

In this paper we propose to work with the challenges like proper initialization of the early stages, designing the states, actions, transitions using Markov Decision Process (MDP) and solving the MDP with two popular reinforcement learning techniques namely Q-learning and SARSA(λ). We also want to compare the convergence speed of these two techniques so that we may conclude about one of them to be better.

In this paper we intend to work with HPE Helion Eucalyptus an open solution for building private and hybrid clouds compatible with Amazon Web Services (AWS) APIs. It can dynamically scale up or down depending on application workloads and is well suited for enterprise clouds. The components that mainly form the Eucalyptus architecture are Cloud Controller (CLC), Cluster Controller (CC), Node Controller (NC), Storage Controller (SC), etc. In our model we include cluster controller, cloud controller and node controller. Cloud Controller is the entry point into the cloud for administrators, project managers, developers or end users. The function of a CLC includes monitoring the availability of resources on various components of the cloud infrastructure, including hypervisor nodes that are used to actually provision the instances and the cluster controllers that manage the hypervisor nodes [4]. Resource arbitration decides which clusters will be used for provisioning the instances, monitoring the running instances. Again, Cluster Controller (CC) generally executes on a cluster front-end machine or any machine that has network connectivity to both the nodes

running NCs and to the machine running the CLC [5]. CCs gather information about a set of VMs and schedules VM execution on specific NCs. The CC also manages the virtual instance network and participates in the enforcement of SLAs as directed by the CLC. All nodes served by a single CC must be in the same broadcast domain (Ethernet). Another component of our model is the Node Controller. Node Controller (NC) is executed on every node that is designated for hosting VM instances. The NC runs on each node and controls the life cycle of instances running on the node [6].

The NC interacts with the OS and the hypervisor running on the node on one side and the CC on the other side. NC queries the operating system running on the node to discover the node's physical resources – the number of cores, the size of memory, and the available disk space [7]. It also learns about the state of VM instances running on the node and propagates this data up to the CC. The function of a node controller is to collect data related to the resource availability and utilization of the node and reporting the data to CC and instance life cycle management. All these components are necessary to be described as the system modeling as Markov Decision Process greatly depends on these components.

## II. Related Work

Adhoc manually determined policies like threshold based policies are used industrially to cope with the VM allocation problem. Low threshold on performance causes more allocation of Virtual machines and a high one causes the reduction of Virtual Machines. Providing good thresholds proved to be tricky and hard to automate to fit every application requirement [8]. Dutreih *et al.* also proposed to solve the VM management problem through machine learning but his action set was limited to only Adding and reducing VMs excluding VM migration and maintenance [9]. In 2015 Enda et al. also tried to solve this same problem with reinforcement learning but the states they introduced did not have clarification of the whole cloud computing architecture [10]. Among the different works on threshold-based policies, Lim et al. propose proportional thresholding to adapt policy parameters at runtime [11]. It consists of modifying the range of thresholds in order to trigger more frequent decisions when necessary. This approach adapts very well to fast changing conditions and is directly integral into automated agents with stability mechanisms [12]. The mentioned paper gives elegant answers to remove latency but when the question comes about to take prompt and efficient decisions for the changing workload patterns it lacks in adaptation. Tesauro *et al.* explore the application of reinforcement learning in a sequential decision making process [13]. The paper presents two novel ideas: the use of a predetermined policy for the initial period of the learning and the use of an approximation of the Q-function as a neural network [14]. The results are interesting, though dependent on the form of the reward function. Besides that, the initial learning with a predetermined policy appears less promising than an initialization using a pre-computing of the Q function through the traditional value-iteration algorithms in a model-based learning approach [15]. Zhang et al. propose a pragmatic approach to resource allocation which consists in pre-allocating enough resources to match up to 95% of the observed workload and then allocates more resources on another cloud when this threshold is passed [16]. This work greatly lacks in real time automatic control approach to outsmart instability. To tell the truth, authors cared about working with the controllers that focused on immediate reward but not stability of the entire system. Most researches that worked upon Q-learning lacks in one thing clearly. We clearly do not know if there are other reinforcement learning algorithms that work better than Q-learning.

## III. Proposed Method

### A. Markov Decision Process

To solve the resource allocation problems in clouds, two types of works have drawn our attention recently that have been used to find the optimal policy. One is threshold based policies that trigger adaptations based on the upper bounds and lower bounds on the performance and another is sequential decision policies based on the MDP models computed using reinforcement learning algorithm [10]. To generate episodic decision making policies for our problem of Virtual Machine Management problem, we propose to use Markov Decision Process (MDP) model. Coarsely speaking, an MDP involves a decision agent that repeatedly observes the current states of the controlled system, takes a decision among the ones allowed in that state and then observes a transition to a new state s′ and a reward r that will drive its decisions [17]. The MDP that models our VM management problem is,

M= {S, A, T, R, β} where:

S= {NC, CC, CLC, w, v, p, $p_{sla}$}

- NC is the Node controller, CC is the Cluster Controller, CLC is the Cloud Controller, w is the Workload, v is the number of virtual machines, p is the performance and $p_{sla}$ is the performance the cloud service provider committed to provide.

- A is the action set that includes adding (a), reducing(r), maintaining (m) and migrating (mg) a virtual machine. Newly arrived VM requests are collected and allocated to physical resources that are not completely used by previously allocated VMs or were freed by the VMs that were de-allocated because their life time expired or tasks completed. This action of allocation is regarded as action "a" in our MDP and on the contrary we consider the action of de-allocation of VMs as action "r" in our MDP. After launching an instance, it goes to a running state. Stopping the instance leads to a stopping state and then stopped state. Again, when an instance is started it goes to a pending state. Rebooting an instance is equivalent to rebooting an OS. The instance remains in the same host computer. An instance is scheduled to be retired when it is detected that the instance has got an irreparable failure of the underlying hardware hosting the instance. The instance

can get terminated when it is necessary or the user wishes to. Our third action "m" is dependent to these sub-actions. VM migration is of three types. Cold migration includes the operation of shutting down VM on a host and restarting it on a new host. Again warm migration includes suspending a VM on a host, copying it across RAM and CPU registers to continue on a new host. Lastly VM live migration includes copying a VM across RAM while VM continues to run. We denote the migration option in our system as "mg".

- T is the probability distribution of going to a state "s′" from "s" by taking any random action "a".
- R is the cost function that expresses the reward if action "a" is taken at state s.
- "β" is the discount factor, 0< β <1.

The following diagram (Fig. 1) describes our MDP:

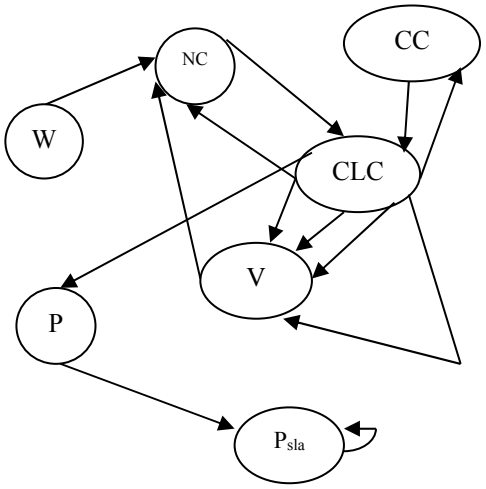


Figure 1. State Diagram

The starting state is NC and the goal state is $P_{sla}$ here. CLC can only go to "v" state by the actions a, r, m, mg. Other states can have transitions as directed through empty (ε) transitions.

The reinforcement learning technique we used here is Q-learning. Q-learning is a model-free reinforcement learning technique. It works by learning an action value function that ultimately gives the expected utility of taking a given action in a given state and following the optimal policy thereafter. Our Q-learning algorithm is [18]:

*B. Q-learning*

1. $(\forall s \in S)(\forall a \in A(s))$;
2. **initialize** Q(s , a)
3. s := the initial observed state
4. **loop**
5. Choose $a \in A(s)$ according to a policy derived from Q
6. Take action a and observe next state s′ and reward r
7. $Q[s , a] := Q[s , a] + \alpha(R[s,a] + \gamma * \max_a Q[s', a'] - Q[s, a])$
8. s := s′
9. **end loop**
10. return $\pi(s) = \text{argmax}_a Q(s , a)$

Here, "α" is the learning rate. It determines how much the old information will be wiped out by the newer one. Value of α being "0" will make the agent not to learn anything and on the contrary value of α being "1" would make it consider only the recent most information. In deterministic environments the value of α can be set to 1 and that is optimal. But our environment is stochastic and it is quite tough to determine the exact value. "$\gamma$" is the discount factor. It determines how important the future rewards can be. A value of "0" will make the agent short sighted and the agent will only consider the current rewards.

*C. SARSA (λ)*

State-Action-Reward-State-Action (SARSA) is another reinforcement algorithm to solve MDP. The name simply reflects that the function that updates the Q value depends on the current state of "s", the action "a", the reward "r" that an agent gets by choosing the action a and the next state "s′". When eligibility traces are added to SARSA algorithm, the algorithm is called SARSA (λ) algorithm [19]. Our SARSA (λ) algorithm is given below [20]:

1. **Initialize** Q(s, a) arbitrarily
2. **Repeat** (for each episode):
3. **Initialize** s
4. Choose a from s using policy derived from Q
5. **Repeat** (for each episode):
6. Take action a, observe r, s′
7. Choose a′ from s′ using policy derived from Q
8. $\boldsymbol{\delta} = r + \gamma Q[s', a'] - Q[s, a]$
9. $e(s, a) = e(s, a)+1$
10. **For** all (s , a):
11. $Q[s , a] = Q[s , a] + \alpha\, \boldsymbol{\delta}\, e(s, a)$
12. $e(s, a) = \gamma\lambda e(s, a)$
13. s=s′ ; a=a′
14. **until** s is terminal

To experiment with the Q-learning and SARSA (λ), we have defined the reward function. It is partially opted from the experiments of Enda Barrett, Enda Howley and Jim Duggan from National University of Ireland Galway [21]. It is as follows:

$$R=\beta\,(\text{Cost}) + (1-\beta)\,(\text{Penalty})$$

Where,

$$\text{Cost} = C_r \times V_a \times (C_r \times V) \quad (1)$$

$$\text{Penalty} = P_c \times (1 + (P_d - P_{sla}) / P_{sla}) \quad (2)$$

In equation (1),

$C_r$=the cost of the resources, the variable depending on the specific configuration and region.

$V_a$ = the specific virtual machine to be added, reduced, maintained and migrated.

V = total number of VMs in the system.

In equation (2),

$P_c$ = penalty for the violation of SLA

$P_d$ = the performance displayed by the system randomly

$P_{sla}$ = target performance

Lastly, β is the balancing factor.

## IV. Experimental Results and Evaluation

We varied the β in accordance with the cost and penalty we acquire in different training episodes and plotted them in a graph while implying Q-learning. We also did the same in case of SARSA (λ). Deciding the value of β was very important as it balances the reverse condition of cost and penalty while working with the reward function. Balancing this reverse condition would also help the cloud providers to meet the service level agreement. Neither the cost can be high nor can the penalty be high. The value of Beta must lead us to the values of cost and penalty that will make the implementation of ours a balanced one. The following table shows us the average (random 10 episodes) of the cost and penalties for different parameters of beta for Q-learning (see Figure 2, Table 1). We can see the vibrant changes of cost and penalty as the values of Beta are varied.

Table1. Cost and Penalty for variant β (Q-learning)

| Value of β | Cost | Penalty |
|---|---|---|
| 0.10 | 2.17 | 6.04 |
| 0.25 | 3.34 | 7.10 |
| 0.50 | 4.84 | 6.90 |
| 0.75 | 2.50 | 7.74 |
| 0.90 | 5.31 | 5.25 |

The graph below shows which value of β balances the cost and Penalty. (Fig.2)

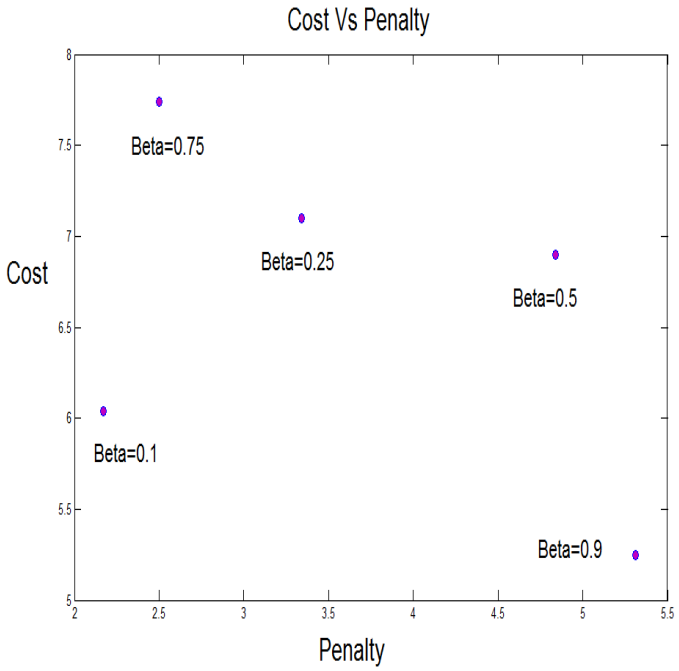


Figure 2. Cost Vs Penalty Graph for beta in Q-learning

Again, the following table shows us the average (random 10 episodes) of the cost and penalties for different parameters of beta for SARSA (λ) (see Figure 3, Table 2).

| Value of β | Cost | Penalty |
|---|---|---|
| 0.10 | 33.21 | 7.21 |
| 0.25 | 71.08 | 6.00 |
| 0.50 | 75.50 | 8.07 |
| 0.75 | 81.61 | 6.39 |
| 0.90 | 90.93 | 6.18 |

The upcoming graph shows which value of β balances the cost and Penalty for SARSA (λ) (see Fig 3).

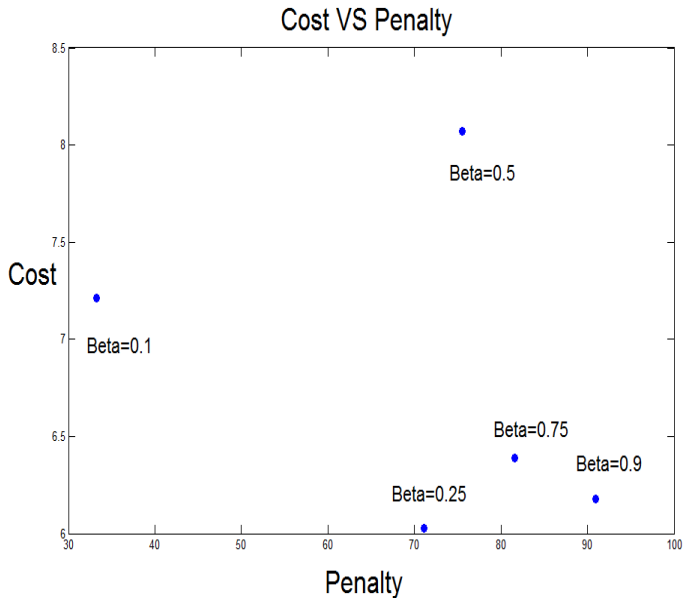


Figure 3. Cost Vs Penalty Graph for beta in SARSA (lambda)

To compare the beta values of these two reinforcement learning techniques, we merged the graphs stated above and observed the versatile values of beta. This helped us greatly to decide which algorithm will give us higher chunks of reward. The graph below shows us the comparison of the beta values for both of the learning techniques (Fig.4)

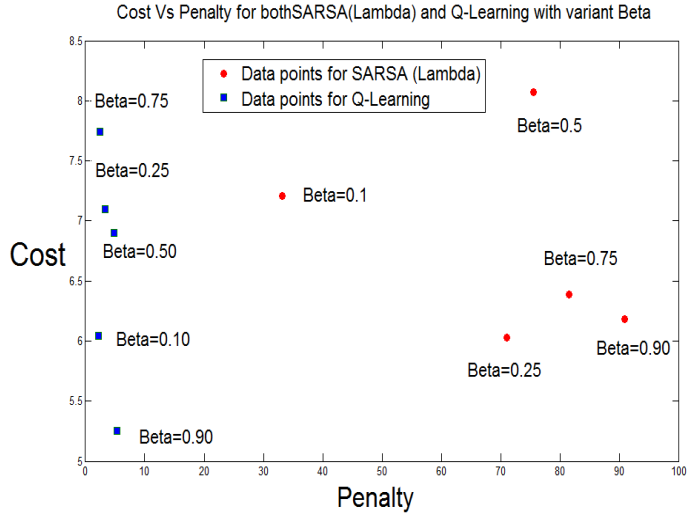


Figure 4. Cost Vs Penalty Graph for beta in Q and SARSA(λ)

For SARSA (λ) algorithm, we also varied the values of lambda that gave us the following result (see Figure 5). This was important to find the value of lambda that will best balance the SARSA (λ) convergent.

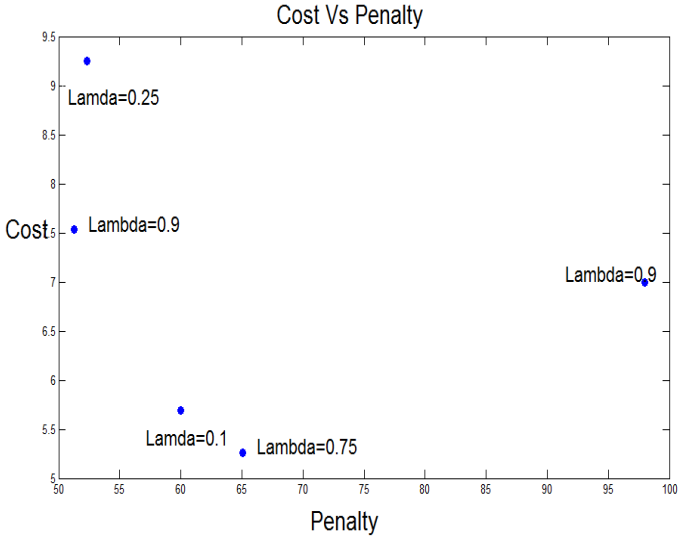


Figure 5. Cost Vs Penalty Graph for λ in SARSA (λ)

To decide up to what extent the newly acquired information will override the old information, learning rate was varied throughout the experiment while applying Q-learning. Alpha having the value of "0" will make the agent not to learn anything at all. On the contrary if the value of alpha is "1" it will only consider the recent most information. The values of alpha that we varied were 0.1, 0.25, 0.5, 0.75 and 0.9. These values generated variant results with rewards. The Alpha was chosen as 0.1 because this the only value of alpha in which the convergence took place (see Fig 6).

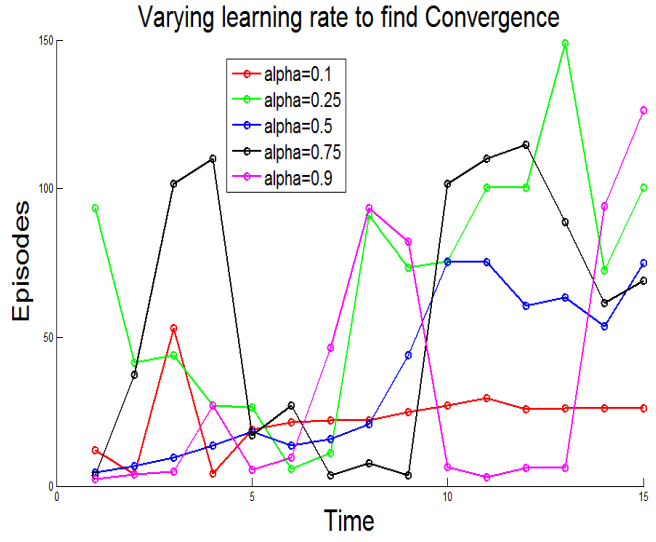


Figure 6. Different values of alpha producing chunks of reward

Learning rate was varied throughout the experiment while applying SARSA (lambda) too. The values of alpha that we varied were 0.1, 0.25, 0.5, 0.75 and 0.9. These values generated variant results with rewards. Alpha was chosen as 0.1 because this is the only value of alpha in which the convergence took place (see Fig 7).

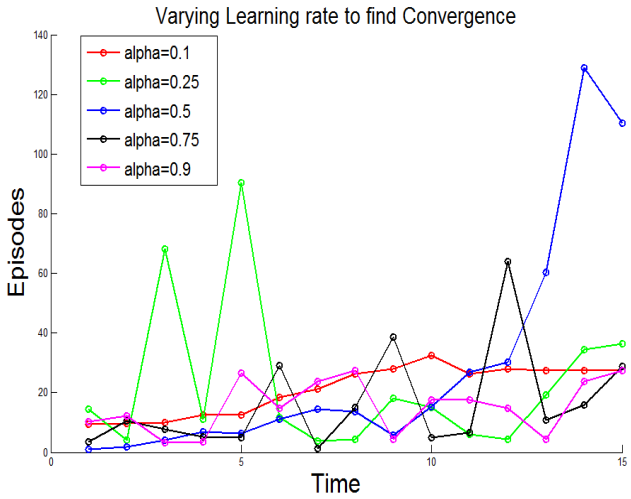


Figure 7. Different values of alpha producing chunks of reward

Lastly, the most important graph for our experiment is the convergence comparison of Q-learning and SARSA (λ) algorithm (see Fig 8). This graph is very important because it can help the researchers come to a decision that which algorithm works better. Q-learning shows better performance comparing with SARSA(lambda) in terms of convergence speed but provides less performance comparing with the cumulative average reward. Faster convergence is very important as time management in VM management problem relates to the service level agreement provided by the cloud providers.

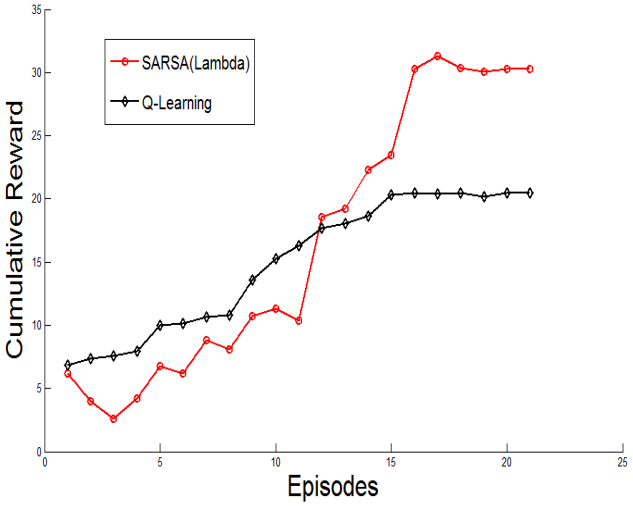


Figure 8. Convergence Comparison of Q and SARSA (λ)

From different values of beta in case of Q-learning, we chose 0.1. It balances the reverse condition of cost and penalty

best. Again, for SARSA (λ) value of beta was chosen 0.25 as it best balanced the contrary propositions of Cost and Penalty. In, SARSA (λ), λ is a very important parameter. It was fixed to 0.9. While comparing the convergence of these two reinforcement learning techniques, we were amazed to see that early convergence took place in case of Q learning and SARSA (λ) converged later. For our case scenario, Q-learning showed the better performance.

## V. Conclusion

We implemented the two reinforcement learning techniques namely Q-learning and SARSA (λ) in real-time Eucalyptus cloud architecture. Previous approaches towards automating Virtual Machine management does not enlighten us with the comparative study regarding which reinforcement learning technique is better to opt for. Again, none of the previous approaches have a system model designed with the components of real time cloud architecture. Our action set consists of the actions of adding, reducing and maintaining the virtual machines. They completely ignored the virtual machine migration part that is a must to include in any action set of MDP considering the issue of virtual machine management. Currently we are working on to implement our model in two of the cloud simulators "CloudSim" and "ICanCloud". Implementing our model with a huge number of nodes will be a great challenge for us in future.

## References


[1] E. Leith, "What Are Basic Differences between IAAS, PAAS and SAAS?"Quora, n.d. Web. 25 Mar. 2016. <https://www.quora.com/What-are-basic-differences-between-IAAS-PAAS-and-SAAS>.

[2] E. Leith, "What Are Basic Differences between IAAS, PAAS and SAAS?"Quora, n.d. Web. 25 Mar. 2016. <https://www.quora.com/What-are-basic-differences-between-IAAS-PAAS-and-SAAS>.

[3] E. Leith, "What Are Basic Differences between IAAS, PAAS and SAAS?"Quora, n.d. Web. 25 Mar. 2016. <https://www.quora.com/What-are-basic-differences-between-IAAS-PAAS-and-SAAS>.

[4] Y. Wadia, "The Eucalyptus Open-Source Private Cloud." *Www.cloudbook.net*. CloudBook, n.d. Web. 25 Mar. 2016. <http://www.cloudbook.net/resources/stories/the-eucalyptus-open-source-private-cloud>.

[5] Y. Wadia, "The Eucalyptus Open-Source Private Cloud." *Www.cloudbook.net*. CloudBook, n.d. Web. 25 Mar. 2016. <http://www.cloudbook.net/resources/stories/the-eucalyptus-open-source-private-cloud>.

[6] Y. Wadia, "The Eucalyptus Open-Source Private Cloud." *Www.cloudbook.net*. CloudBook, n.d. Web. 25 Mar. 2016. <http://www.cloudbook.net/resources/stories/the-eucalyptus-open-source-private-cloud>.

[7] Y. Wadia, "The Eucalyptus Open-Source Private Cloud." *Www.cloudbook.net*. CloudBook, n.d. Web. 25 Mar. 2016. <http://www.cloudbook.net/resources/stories/the-eucalyptus-open-source-private-cloud>.

[8] X. Dutreilh, N. Rivierre, A. Moreau, J. Malenfant, and I. Truck, "From Data Center Resource Allocation to Control Theory and Back," inProc. of the 3rd IEEE Int. Conf. on Cloud Computing, CLOUD 2010, application and industry track. IEEE, 2010, pp. 410–417.

[9] X. Dutreilh, N. Rivierre, A. Moreau, J. Malenfant, and I. Truck, "From Data Center Resource Allocation to Control Theory and Back," inProc. of the 3rd IEEE Int. Conf. on Cloud Computing, CLOUD 2010, application and industry track. IEEE, 2010, pp. 410–417.

[10] E. Barrett, E. Howley and J. Duggan, "Applying Reinforcement Learning Towards Automating Resource Allocation and Application Scalability in the Cloud." *Applying Reinforcement Learning Towards Automating Resource Allocation and Application Scalability in the Cloud*(2011): 1-18. Web. 12 June 2015.

[11] H. C. Lim, S. Babu, and J. S. Chase, "Automated control for elastic storage," in Proc. of the 7th Int. Conf. on Autonomic computing (ICAC). ACM, 2010, pp. 1–10.

[12] X. Dutreilh, N. Rivierre, A. Moreau, J. Malenfant, and I. Truck, "From Data Center Resource Allocation to Control Theory and Back," inProc. of the 3rd IEEE Int. Conf. on Cloud Computing, CLOUD 2010, application and industry track. IEEE, 2010, pp. 410–417.

[13] G. Tesauro, N. K. Jong, R. Das, and M. N. Bennani, "A Hybrid Reinforcement Learning Approach to Autonomic Resource Allocation," in Proc. of the 2006 IEEE Int. Conf. on Autonomic Computing (ICAC). IEEE Computer Society, 2006, pp. 65–73.

[14] X. Dutreilh, N. Rivierre, A. Moreau, J. Malenfant, and I. Truck, "From Data Center Resource Allocation to Control Theory and Back," inProc. of the 3rd IEEE Int. Conf. on Cloud Computing, CLOUD 2010, application and industry track. IEEE, 2010, pp. 410–417.

[15] M. L. Littman, "Algorithms for Sequential Decision Making," Ph.D. dissertation, Dep. of Computer Science, Brown U., mars 1996.

[16] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, and A. Saxena, "Resilient workload manager: taming bursty workload of scaling internet applications," in Proc. of the 6th Int. Conf. industry session on Autonomic computing and communications. ACM, 2009, pp. 19–28.

[17] X. Dutreilh, N. Rivierre, A. Moreau, J. Malenfant, and I. Truck, "From Data Center Resource Allocation to Control Theory and Back," inProc. of the 3rd IEEE Int. Conf. on Cloud Computing, CLOUD 2010, application and industry track. IEEE, 2010, pp. 410–417.

[18] R. S. Sutton and A. G. Barto, *Reinforcement* Learning: An Introduction. The MIT Press, Cambridge, Massachusetts, England, 2002

[19] K. M. Gupta. "Performance Comparison of Sarsa(λ) and Watkin's Q(λ) Algorithms." (n.d.): n. pag. Print.

[20] R. S. Sutton and A. G. Barto. *Reinforcement* Learning: An Introduction. The MIT Press, Cambridge, Massachusetts, England, 2002

[21] E. Barrett, E. Howley and J. Duggan, "Applying Reinforcement Learning Towards Automating Resource Allocation and Application Scalability in the Cloud." *Applying Reinforcement Learning Towards Automating Resource Allocation and Application Scalability in the Cloud*(2011): 1-18. Web. 12 June 2015.




View publication stats