Reinforcement Learning based Autonomic Virtual Machine Management in Clouds

School of Computer Science and Engineering, Dhaka

Bangladesh

By

Md. Arafat Habib

Under the supervision of

Dr. Md. Muhidul Islam Khan

Thesis Submitted in partial fulfillment of the requirement for the degree of

Bachelor of Science

In

Computer Science and Engineering

Under the supervision of

Dr. Md. Muhidul Islam Khan

By

Md. Arafat Habib (ID: 12101056)

April, 2016

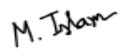
TABLE OF CONTENTS

DECLARATION

thesis titled We do hereby declare that the“Reinforcement Learning based Autonomic in Clouds Virtual Machine Management” is submitted to the Department of Computer Science and Engineering of BRAC University in partial fulfillment of the completion of Bachelors of Science in Computer Science and Engineering. We hereby declare that this thesis is based on results obtained from our own work. Due acknowledgement has been made in the text to all other material used. This thesis, neither in whole nor in part has been previously submitted to any University or Institute for the award of any degree or diploma. The materials of work found by other researchers and sources are properly acknowledged and mentioned by reference.

Dated: April 18, 2016

Signature of Supervisor



Signature of Authors

Dr.Md.Muhidul Islam Khan

Assistant Professor

Department of Computer Science and Engineering,

BRAC University

Dhaka, Bangladesh

Arafat Habib (12101056)

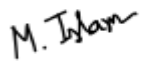
FINAL READING APPROVAL

Thesis Title:

Reinforcement Learning based Autonomic Virtual Machine Management in Clouds

Date of Submission: April 18, 2016

Signature of Supervisor



Dr.Md.Islam Khan Muhidul

Assistant professor

Department of Computer Science and Engineering

BRAC University

Dhaka, Bangladesh

Acknowledgement

Numerous people have supported us during the development of this dissertation. A few words' mention here cannot adequately capture all my appreciation.

I am very thankful to my thesis coordinator Dr.Md.Muhidul Islam Khan, Assistant Professor, Department of Computer Science and Engineering, BRAC University for guiding me throughout my thesis work. Without his key contributions, it would have been quite impossible to finish the work.

I would also like to thank Mr. Kunuk Nykjær from IT University of Copenhagen for his assistance.

Lastly, all the credits go to almighty for making us successful.

Date: April 18, 2016

Md. Arafat Habib

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 thesis, 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).

List of Figures

Figures Page

CHAPTER 1

1. INTRODUCTION

1.1Motivation

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 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 [1]. Auto Scaling, 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

1.2 Goal

In this thesis, 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.

1.3 Thesis Layout

The rest of the thesis is organized as follows:

Chapter 2 describes the Related work, Chapter 3 discusses about the fundamental concepts of cloud computing and virtualization technology, Chapter 4 describes architecture of HPE Helion Eucalyptus cloud, Chapter 5 talks about the proposed system model of ours, Chapter 6 enlightens the proposed method, Chapter 7 presents the experimental results and lastly Chapter 8 concludes the paper with a summary of our work and future works that can be accomplished on our work.

CHAPTER 2

2. Related Works

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 [32Dutreih *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 [2]. 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 [3]. Among the different works on threshold-based policies, Lim et al. propose proportional thresholding to adapt policy parameters at runtime [4]. It consists in 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 [2]. 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 process [5]. 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 [2]. 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 amodel-based learning approach [6]. 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 [7]. 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.

CHAPTER3

3. CLOUD COMPUTING

3.1. General Concept:

Cloud computing is a relatively new model in the computing world. According the definition of National Institute of Standards and Technology, Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. A cloud infrastructure is the collection of hardware and software that enables the five essential characteristics of cloud computing. The cloud infrastructure can be viewed as containing both a physical layer and an abstraction layer. The physical layer consists of the hardware resources that are necessary to support the cloud services being provided, and typically includes server, storage and network components. The abstraction layer consists of the software deployed across the physical layer, which manifests the essential cloud characteristics. Conceptually the abstraction layer sits above the physical layer [8].This cloud model is composed of five essential characteristics, three service models, and four deployment models. The essential characteristics are namelyOn-demand self-service, Broad network access, Resource pooling, Rapid elasticity and Measured service.

On-demand self-service:

A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider [8].

Broad network access:

Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations) [8].

Resource pooling:

The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth [8].

Rapid elasticity:

Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time [8].

Measured service:

Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service [8].

Once a cloud is established, the method of its cloud computing services deployment in terms of business models can differ depending on requirements. The primary service models being deployed are commonly known as:

SAAS:

Software as a Service model is based on multi-tenant architecture. This model enables all customers (tenants) to use single version application with single configuration. To avoid conflicts and provide scalability, application is installed on multiple machines. In some cases, SaaS do not use multi-tenancy. They use other mechanisms such as virtualization where a large number of customers are managed in place of multi-tenancy. Some SaaS solutions do not use multi-tenancy, or use other mechanisms-such as virtualization-to cost-effectively manage a large number of customers in place of multi-tenancy.

PAAS:

Platform-as-a-Service provides a computing platform and solution stack as a service. In this model user or consumers creates software using tools or libraries from the providers. Consumer also controls software deployment and configuration settings. Main aim of provider is to provide networks, servers, storage and other services. PaaS offers deployment of applications by reducing the cost and complexity of buying and maintaining hardware and software and provisioning hosting capabilities. There are various types of PaaS vendors which offer application hosting and a deployment environment along with various integrated services. The services offer scalability and maintenance.

IAAS:

Infrastructure is the foundation of cloud computing. It provides delivery of computing as a shared service reducing the investment cost, operational and maintenance of hardware. Infrastructure should be reliable and flexible for easy implementation and operations of applications. Although the definitions vary widely, all share the common theme of programmatic access to the basic building blocks of IT: compute, storage and networking.

The following picture shows the pyramid hierarchy of different service models in cloud computing [9]:

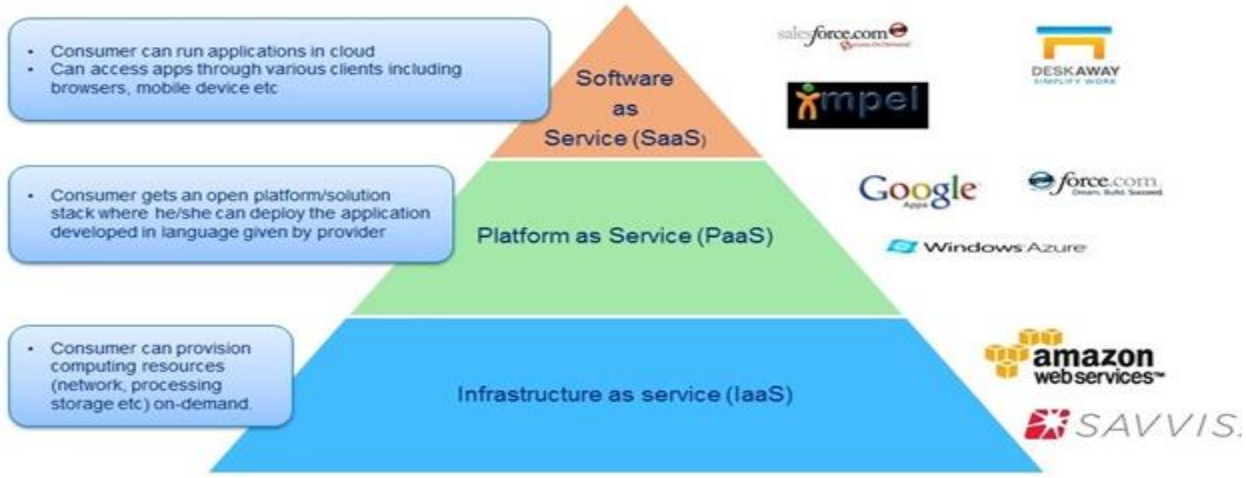


Fig 1: Pyramid view of service model stack of cloud computing

Again, the following figure gives us the user views of different stacks in cloud computing service models [10]:

Fig 2: Users of different service models

The four deployment models that is very common in the field of cloud computing are Private cloud, Community cloud, Public cloud and hybrid cloud.

Private Cloud:
The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

Community Cloud:
The cloud infrastructure is provisioned for exclusive use by a specific communityof consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

Public Cloud:
The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

Hybrid Cloud:
The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds)

The following diagram describes different service models, deployment models and essential characteristics together in cloud architecture [10].

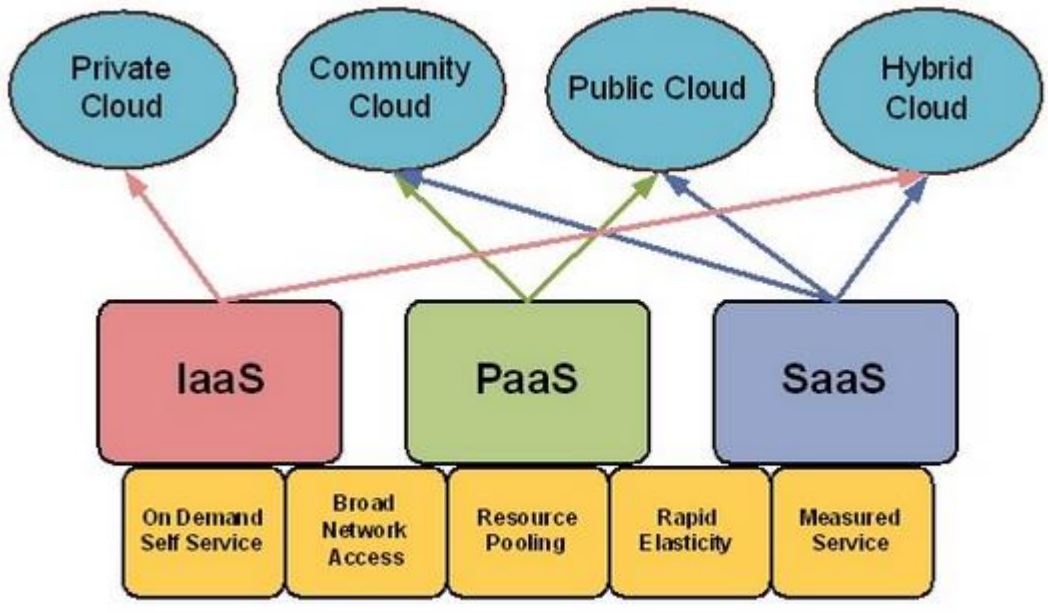


Fig 3: Service models, Deployment models and Essential characteristics together in cloud architecture.

3.2. CLOUD COMPUTING AND VIRTUALIZATION:

Any discussion on cloud computing typically begins with virtualization. Virtualization relates the use of software and hardware for creating the idea that one or more entities related to computing resources exist although the entities in actually, are not physically present. Using virtualization we can take one server appear to be many, desktop computer appear to be running multiple operating system simultaneously or a vast amount of disk space or drives to be available [11]. The most common forms of virtualization include server virtualization, desktop virtualization, virtual networks, virtual storage, etc. Virtualization is mainly using computer resources to imitate other computer resources or whole computers. Judith Hurwitz, Robin Bloor, Marcia Kaufman and Fern Halper in their article “Characteristics of Virtualization in Cloud Computing” discussed

elaborately about the characteristics of Virtualization Technology in cloud computing. Virtualization has three characteristics that make it ideal for cloud computing:

- **Partitioning:** In virtualization, many applications and operating systems (OSes) are supported in a single physical system by *partitioning* (separating) the available resources.
- **Isolation:** Each virtual machine is isolated from its host physical system and other virtualized machines. Because of this isolation, if one virtual-instance crashes, it doesn't affect the other virtual machines. In addition, data isn't shared between one virtual container and another.
- **Encapsulation:** A virtual machine can be represented (and even stored) as a single file, so you can identify it easily based on the service it provides. In essence, the encapsulated process could be a business service. This encapsulated virtual machine can be presented to an application as a complete entity. Therefore, encapsulation can protect each application so that it doesn't interfere with another application.

Virtualization can be applied broadly to just about everything that we could imagine:

- Memory
- Networks
- Storage
- Hardware
- Operating systems
- Applications

What makes virtualization so important for the cloud is that it decouples the software from the hardware. Decoupling means that software is put in a separate container so that it's isolated from operating systems.

To understand how virtualization helps with cloud computing, we must understand its many forms. In essence, in all cases, a resource actually emulates or imitates another resource. Here are some examples:

- **Virtual memory:** Disks have a lot more space than computer memory. Therefore, with virtual memory, the computer frees valuable memory space by placing information it doesn't use often into disk space. PCs have *virtual memory*, which is a disk area that's

used like memory. Although disks are very slow in comparison with memory, the user may never notice the difference, especially if the system does a good job of managing virtual memory. The substitution works surprisingly well.

- **Software:** Companies have built software that can emulate a whole computer. That way, one computer can perform as though it were actually 20 computers. The application consolidation results can be quite significant. For example, you might be able to move from a data center with thousands of servers to one that supports as few as a couple of hundred. This reduction results in less money spent not only on computers, but also on power, air conditioning, maintenance, and floor space.

3.3. Hypervisors:

Bill Kleyman in his article "Hypervisor 101: Understanding the Virtualization Market" talks about the virtualization technology elaborately. The evolution of virtualization greatly revolves around one piece of very important software. This is the *hypervisor*. As an integral component, this software piece allows for physical devices to share their resources amongst virtual machines running as guests on to top of that physical hardware. To further clarify the technology, it's important to analyze a few key definitions:

- **Type I Hypervisor**. This type of hypervisor (pictured at the beginning of the article) is deployed as a bare-metal installation. This means that the first thing to be installed on a server as the operating system will be the hypervisor. The benefit of this software is that the hypervisor will communicate directly with the underlying physical server hardware. Those resources are then *paravirtualized* and delivered to the running VMs.This is the preferred method for many production systems.
- **Type II Hypervisor.** This model (shown below) is also known as a hosted hypervisor. The software is not installed onto the bare-metal, but instead is loaded on top of an already live operating system. For example, a server running Windows Server 2008R2 can have VMware Workstation 8 installed on top of that OS. Although there is an extra hop for the resources to take when they pass through to the VM – the latency is minimal and with today's modern software enhancements, the hypervisor can still perform optimally.

- **Guest Machine.** A guest machine, also known as a *virtual machine (VM)* is the workload installed on top of the hypervisor. This can be a virtual appliance, operating system or other type of virtualization-ready workload. This guest machine will, for all intents and purposes, believe that it is its own unit with its own dedicated resources. So, instead of using a physical server for just one purpose, virtualization allows for multiple VMs to run on top of that physical host. All of this happens while resources are intelligently shared between other VMs.
- **Host Machine.** This is known as the physical host. Within virtualization, there may be several components – SAN, LAN, wiring, and so on. In this case, we are focusing on the resources located on the physical server. The resource can include RAM and CPU. These are then divided between VMs and distributed as the administrator sees fit. So, a machine needing more RAM (a domain controller) would receive that allocation, while a less important VM (a licensing server for example) would have fewer resources. With today's hypervisor technologies, many of these resources can be dynamically allocated.
- **Paravirtualization Tools.** After the guest VM is installed on top of the hypervisor, there usually is a set of tools which are installed into the guest VM. These tools provide a set of operations and drivers for the guest VM to run more optimally. For example, although natively installed drivers for a NIC will work, paravirtualized NIC drivers will communicate with the underlying physical layer much more efficiently. Furthermore, advanced networking configurations become a reality when paravirtualized NIC drivers are deployed.

Modern computer systems are complex structures containing numerous closely interacting components in both software and hardware. Within this universe, virtualization acts as a type of interconnection technology [12]. Interjecting virtualizing software between abstraction layers near the HW/SW interface forms a virtual machine that allows otherwise incompatible subsystems to work together[12]. Further, replication by virtualization enables more flexible and efficient use of hardware resources.

CHAPTER 4

4.HPE Helion Eucalyptus

In our research, 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 from the Eucalyptus architecture are Cloud Controller, Cluster Controller, Node Controller, Storage Controller, 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 [13]. Resource arbitration–deciding 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 [13]. 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 [13].

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. 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 on 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. The following figure describes the fundamental building blocks of Eucalyptus cloud architecture:

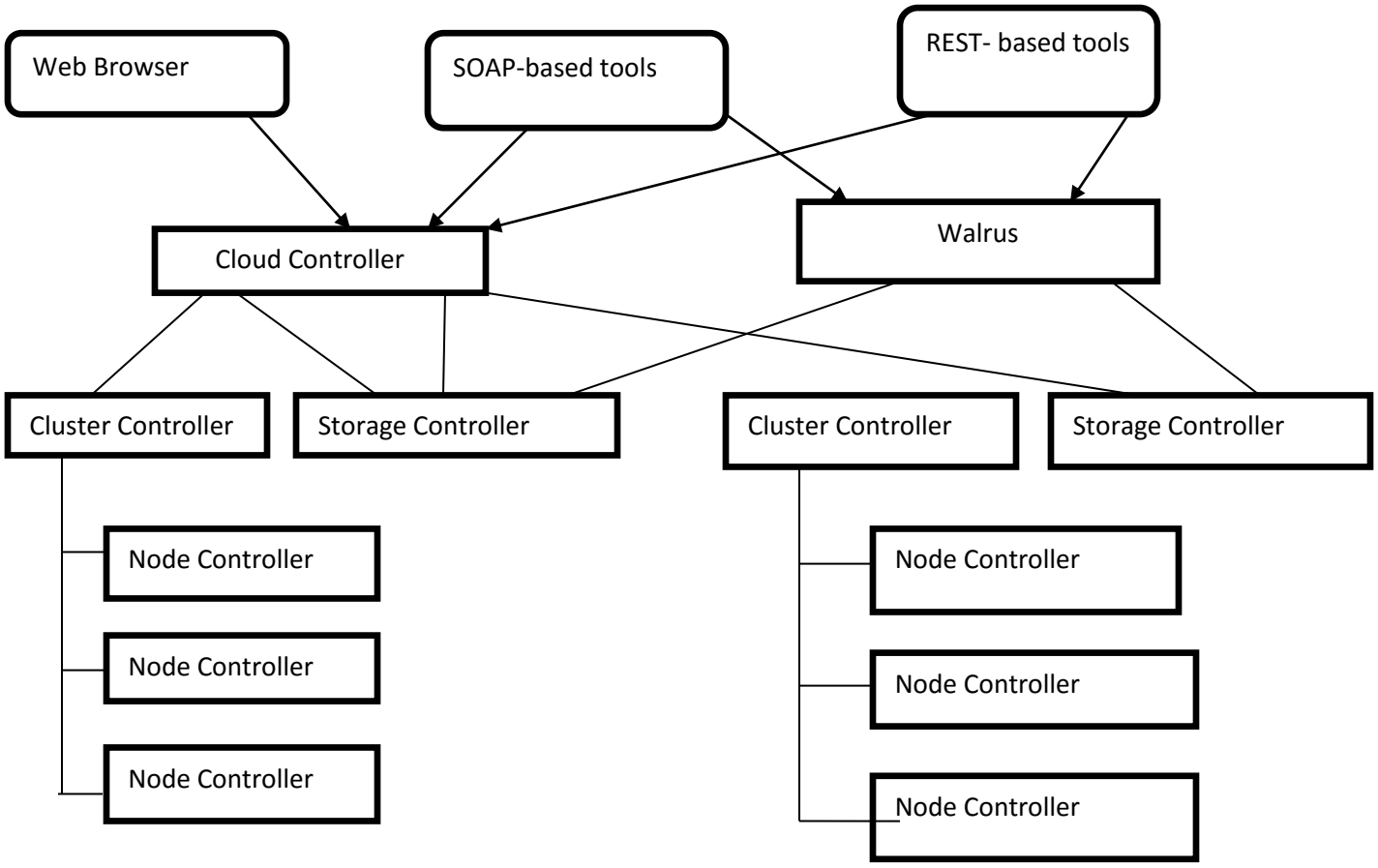


Fig 4: Eucalyptus Cloud architecture with basic components

To understand the hierarchical view of this architecture, we can have a glance at the following figure too:

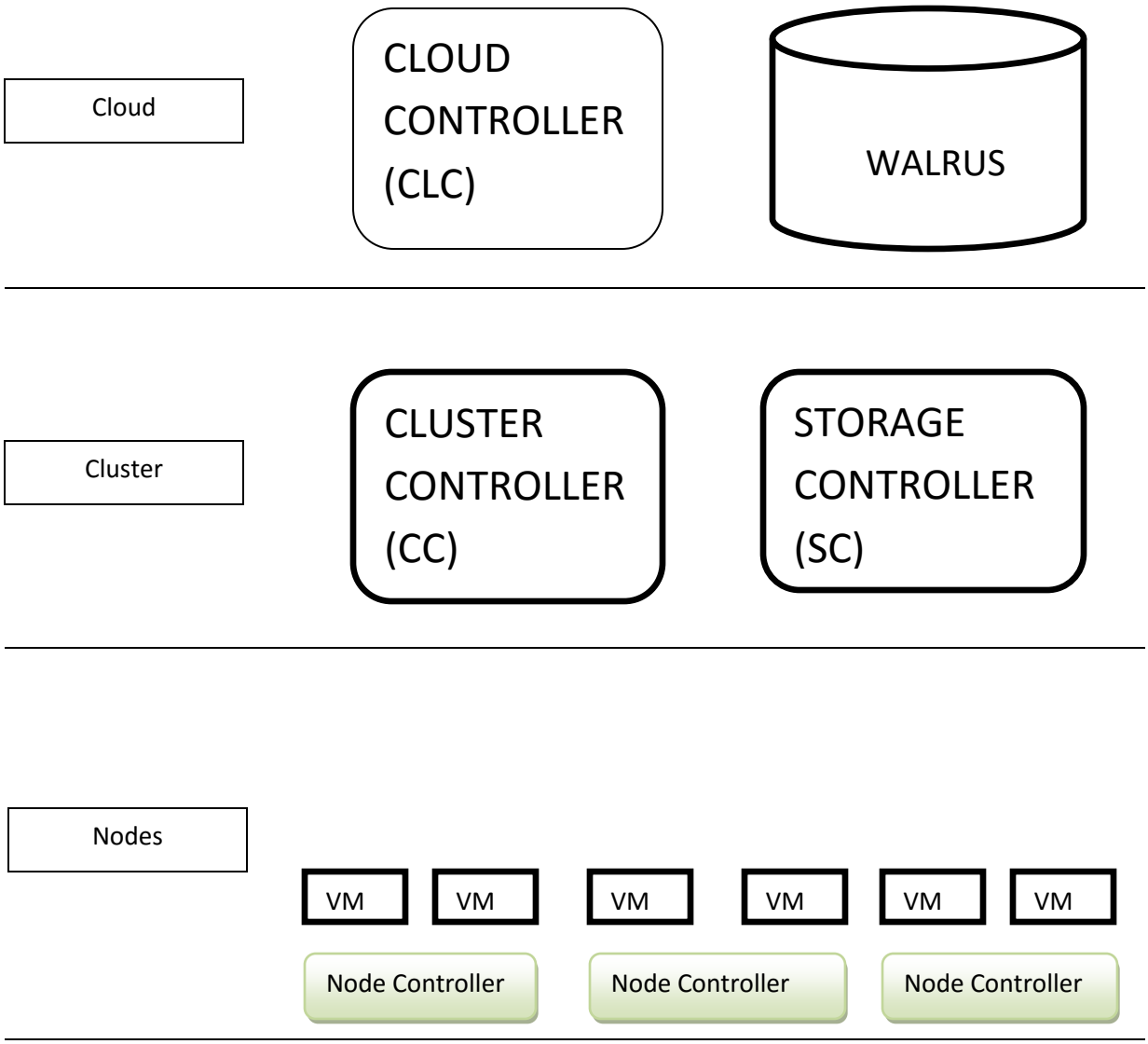


Fig 5: Hierarchical view of Eucalyptus

Now, question arises how the deployment models are connected with the Eucalyptus components. The physical diagram given below shows us the required connectivity [14]:

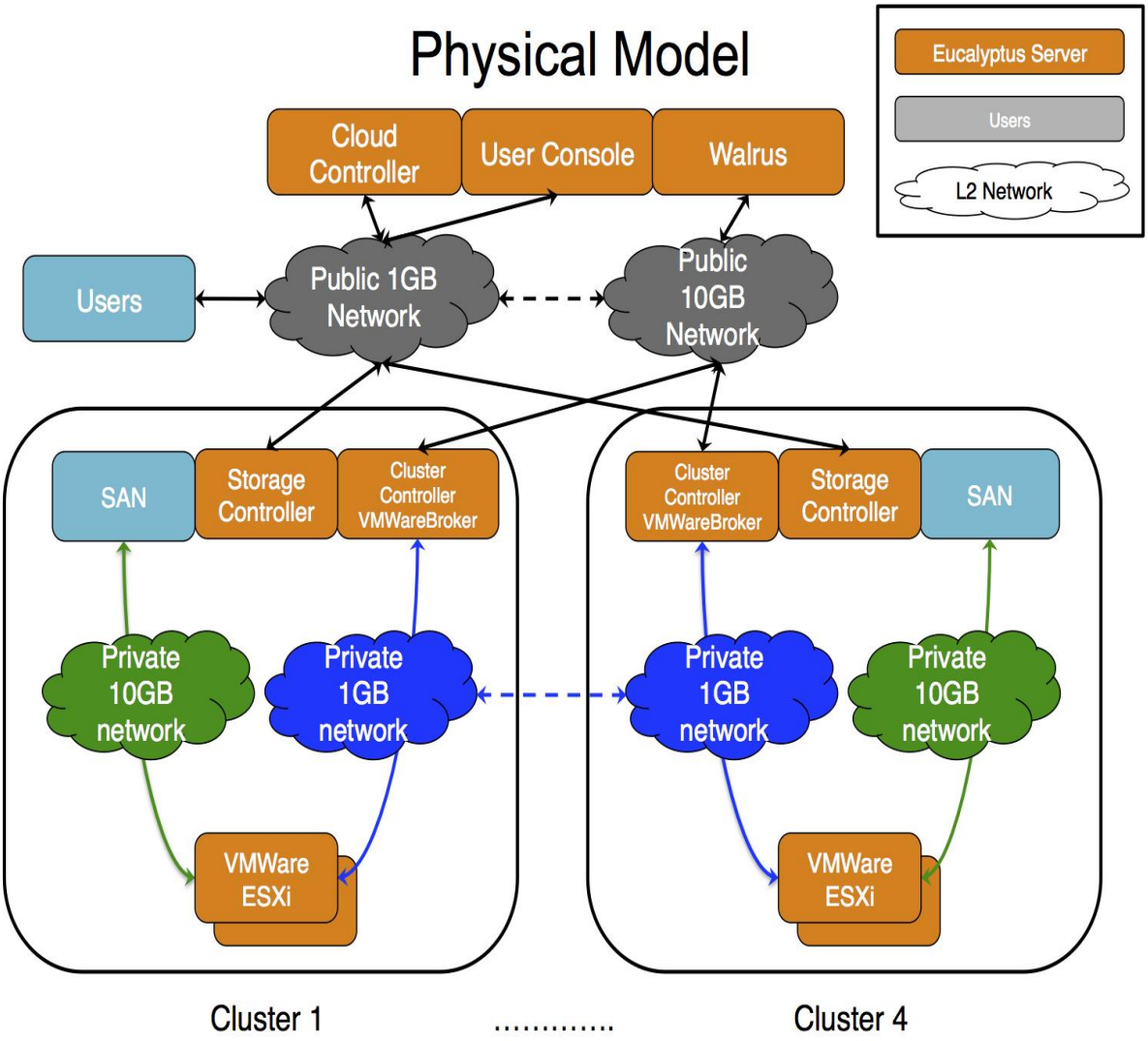


Fig 6: The physical diagram of Eucalyptus cloud

CHAPTER 5

5. Proposed System Model

5.1. Markov Decision Process:

If we consider the VM allocation problem as a decision making problem, it requires regular observance of workload, number of allocated VMs, amount of waiting time in seconds while processing a request. To generate sequential decision making policies for this problem of VM allocation we are to use Markov Decision Process (MDP) model and the computation is to be done by reinforcement learning. An MDP has a decision agent to repeatedly and continuously observe the current state of the system. After the close observation it takes a decision that is allowed to be taken in that state and then observes a transition to a new state. A reward influences the decisions of the agent.

An MDP model contains:

1. A set of possible states S

2. A set of possible actions A

3. A real valued reward function R(s, a)

4. A description T of each action's effects in each state.

5. Stochastic actions:

T: S ×A → Prob(S), for each state and action we specify a new Probability distribution over next states. Representation of the distribution is P(s ′ |s, a).

To solve our resource allocation problems in clouds, two types of works have drawn our attention recently that has 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 Markovian Decision Processes (MDP) models

computed using reinforcement learning algorithm [2]. 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 [28]. The MDP that models our VM management problem is,

M= {S, A, T, R, β} where:

S= {NC, CC, CLC, w, v, p, psla}

- 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 psla 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 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, $0 < \beta < 1$.

The following diagram describes our MDP:

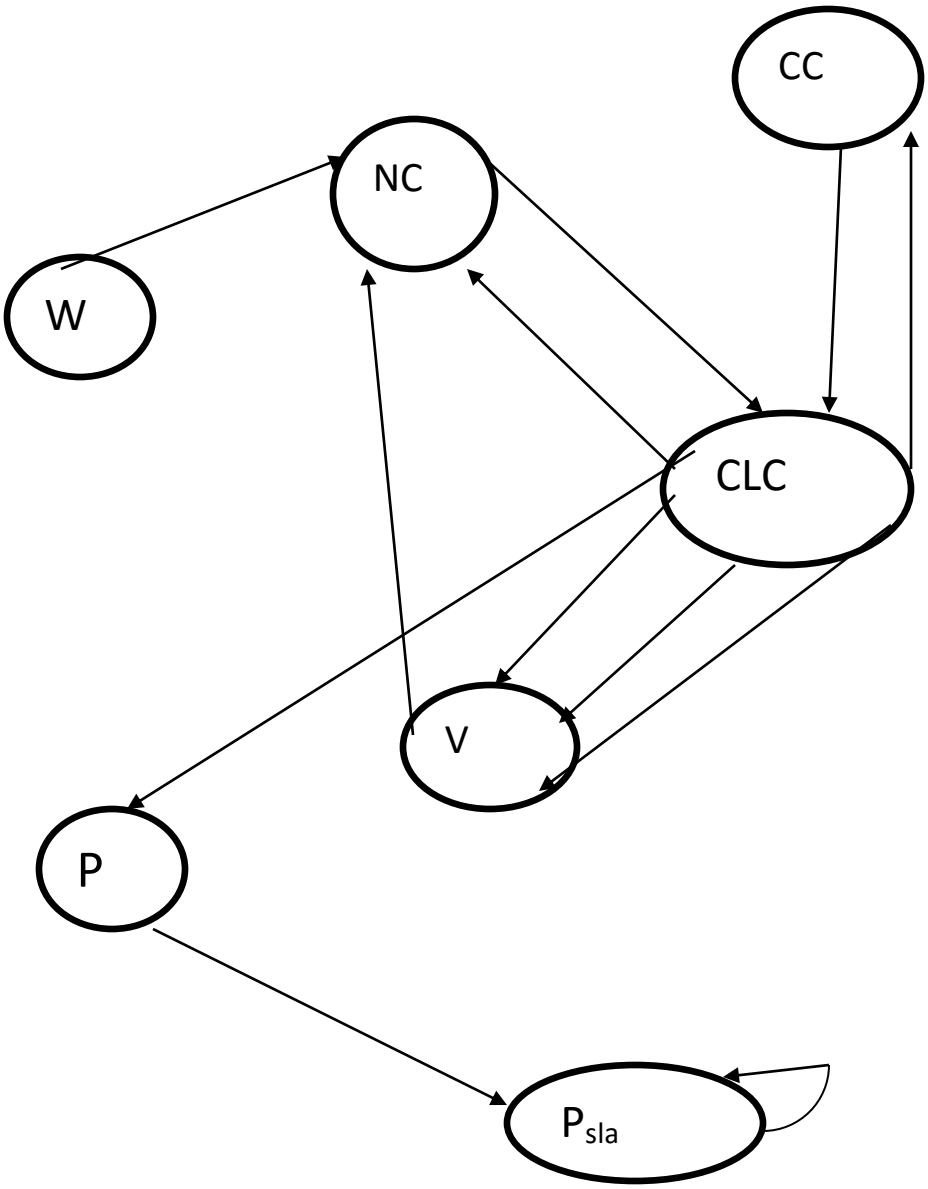


Fig7: State Diagram

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.

CHAPTER 6

6. PROPOSED METHOD

6.1 Q-learning

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 [15]:

A. 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 ∈ 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 to 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. "γ" 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.

6.2 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 [16]. Our SARSA (λ) algorithm is given below [15]:

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. **δ** = r+ γ 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] + α **δ** e(s, a)
12. e(s, a) = γλe(s, a)
13. s=s′ ; a=a′
14. **until** s is terminal

Eligibility trace is a very important term in SARSA (λ) algorithm. There are two ways to view eligibility traces. The more theoretical view, which we emphasize here, is that they are a bridge from TD to Monte Carlo methods. When TD methods are augmented with eligibility traces, they produce a family of methods spanning a spectrum that has Monte Carlo methods at one end and one-step TD methods at the other. In between are intermediate methods that are often better than either extreme method. In this sense eligibility traces unify TD and Monte Carlo methods in a valuable and revealing way.

The other way to view eligibility traces is more mechanistic. From this perspective, an eligibility trace is a temporary record of the occurrence of an event, such as the visiting of a state or the taking of an action. The trace marks the memory parameters associated with the event as eligible for undergoing learning changes. When a TD error occurs, only the eligible states or actions are assigned credit or blame for the error. Thus, eligibility traces Help Bridge the gap between

events and training information. Like TD methods themselves, eligibility traces are a basic mechanism for temporal credit assignment.

6.3 Reward Function

To experiment with the Q-learning and SARSA (λ), we have defined the reward function that has been used is as follows:

$R=\beta\ (Cost) + (1-\beta)\ (Penalty)$

Where,

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

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

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.

CHAPTER 7

7. EXPERIMENTAL RESULTS

7.1 Variant Beta (β)

We varied the β in accordance with the cost and penalty we acquire in different training episodes and plot them in a graph while implying q-learning. We also did the same in case of SARSA (λ). 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 1, Table 1).

Table1. Cost and Penalty for variant beta(Q)

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:

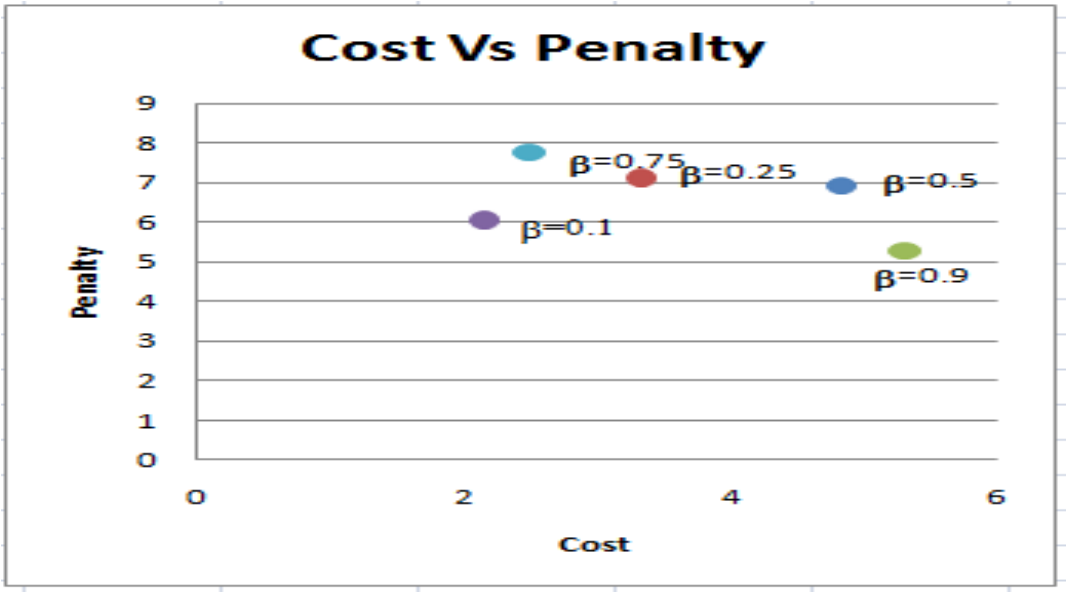


Figure: 8 Cost Vs. Penalty Graph for beta in Q -learning

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

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 graph below shows which value of β balances the cost and Penalty for SARSA (λ):

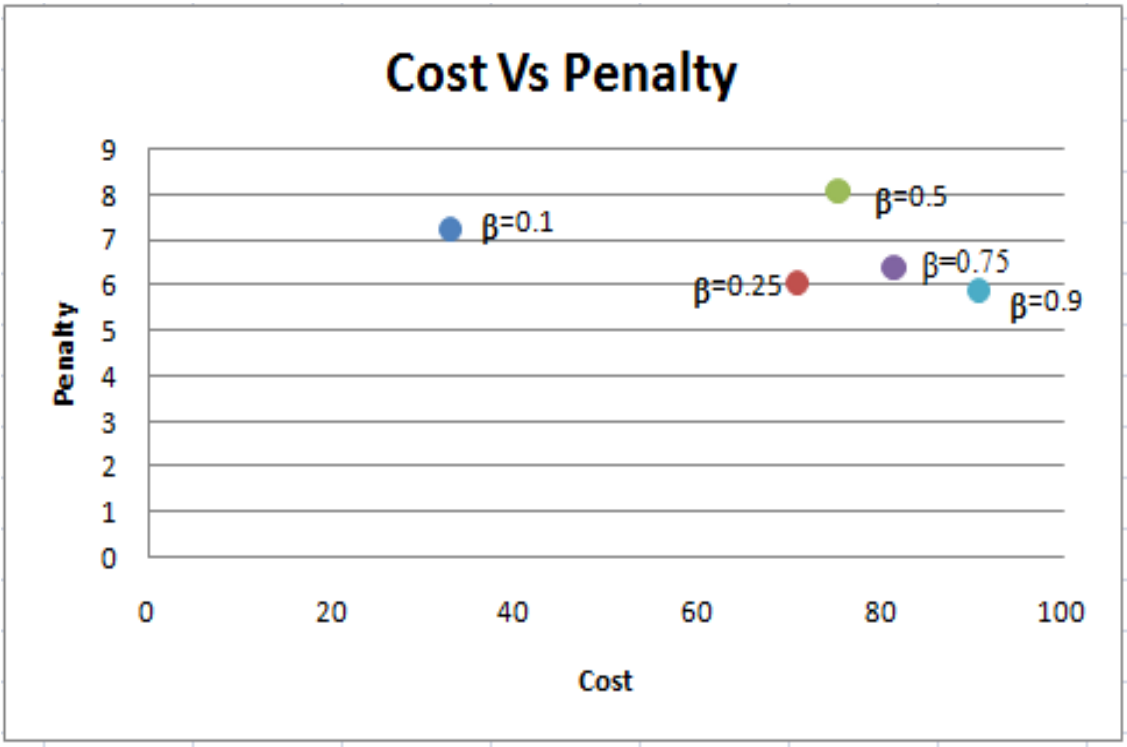


Fig 9:Cost Vs. Penalty Graph for beta in SARSA- λ

To compare the beta values of these two reinforcement learning techniques, we merged the graphs stated above and observed the versatile values of beta. The graph below shows us the comparison of the beta values for both of the learning techniques:

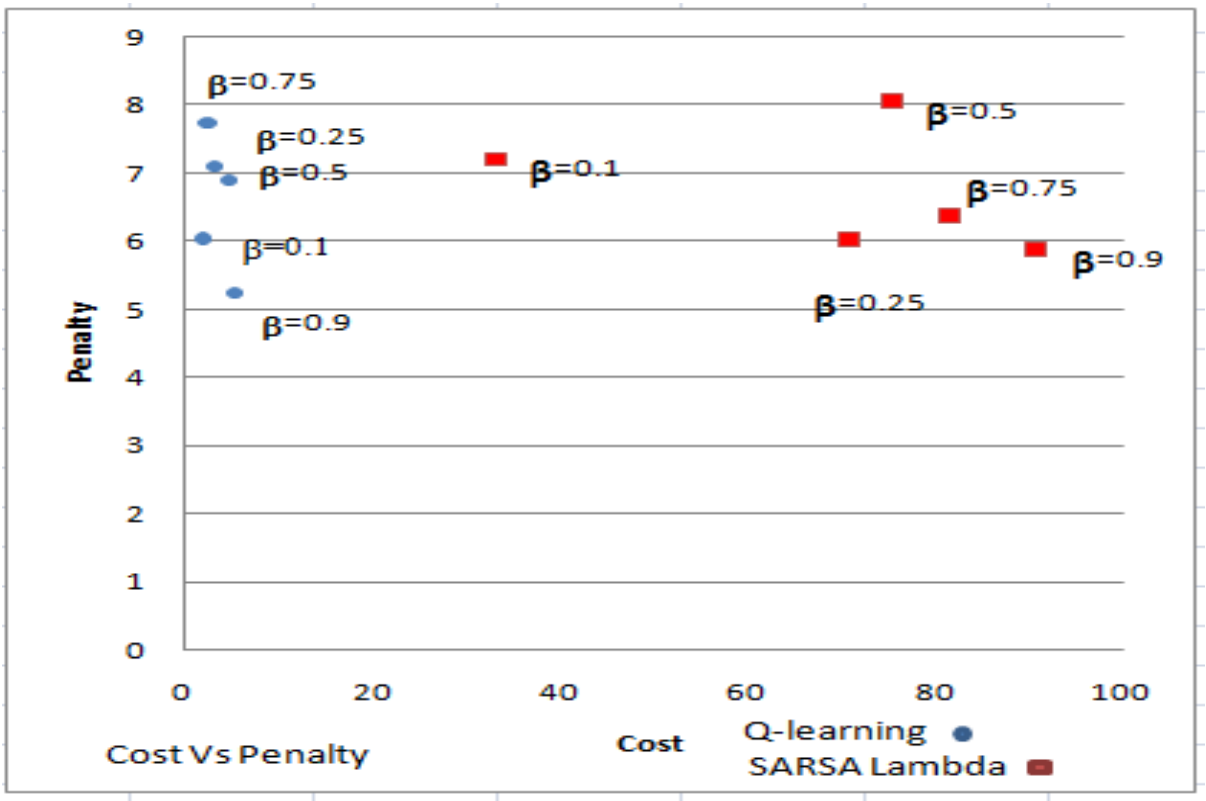


Fig 10: Cost Vs. Penalty Graph for beta in Q and SARSA(λ)

7.2 Variant Lambda (λ)

For SARSA (λ) algorithm, we also varied the values of lambda to see which value of lambda best balances the reverse condition between cost and penalty. The values of lambda taken on account are 0.1, 0.25, 0.5, 0.75, and 0.9. It gave us the following result:

Value of λ	Cost	Penalty
0.1	60.01	5.69
0.25	52.29	9.25
0.5	97.98	7.0
0.75	65.08	5.26
0.9	51.29	7.53

The values gave us the following result:

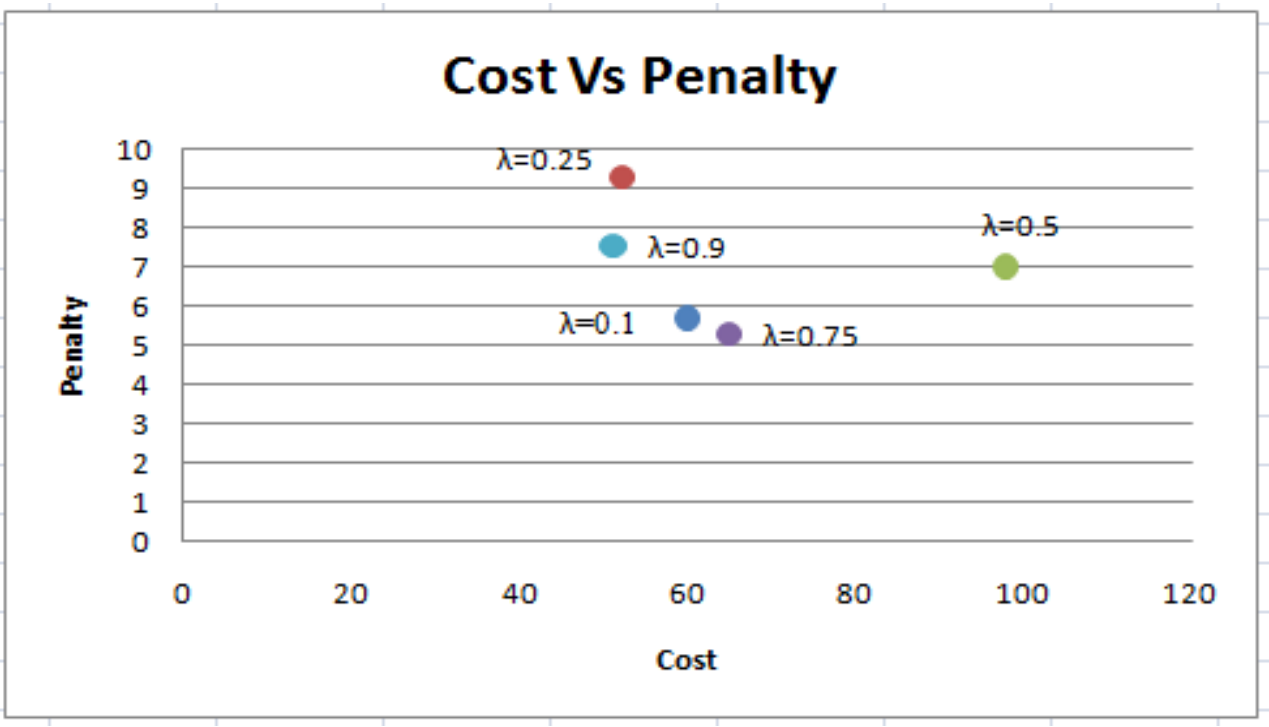


Fig 11: Cost Vs. Penalty Graph for λ in SARSA(λ)

7.3 Variant Alpha (α)

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. 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. The following graph represents different values of alpha generating chunks of reward:

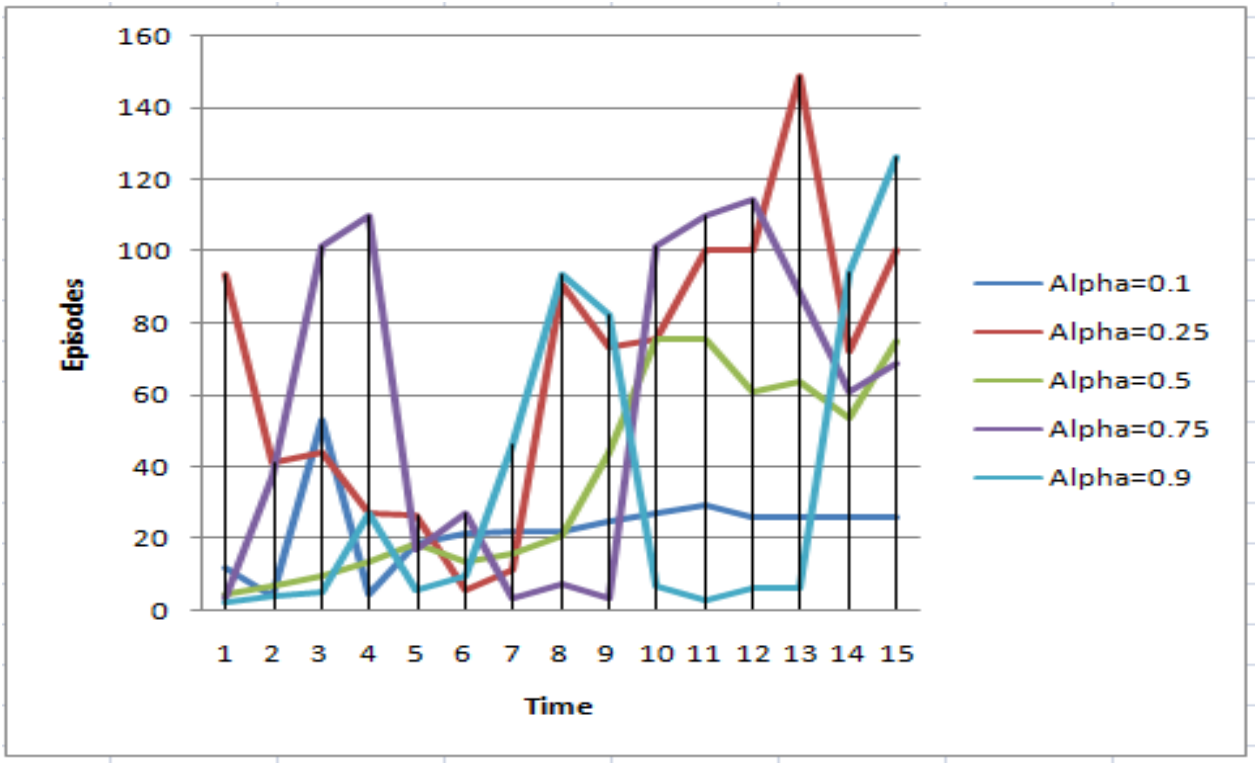


Fig 12: Different values of alpha producing chunks of reward

Learning rate was varied throughout the experiment while applying SARSA-lamda 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 the is the only value of alpha in which the convergence took place. The following graph represents different values of alpha generating chunks of reward:

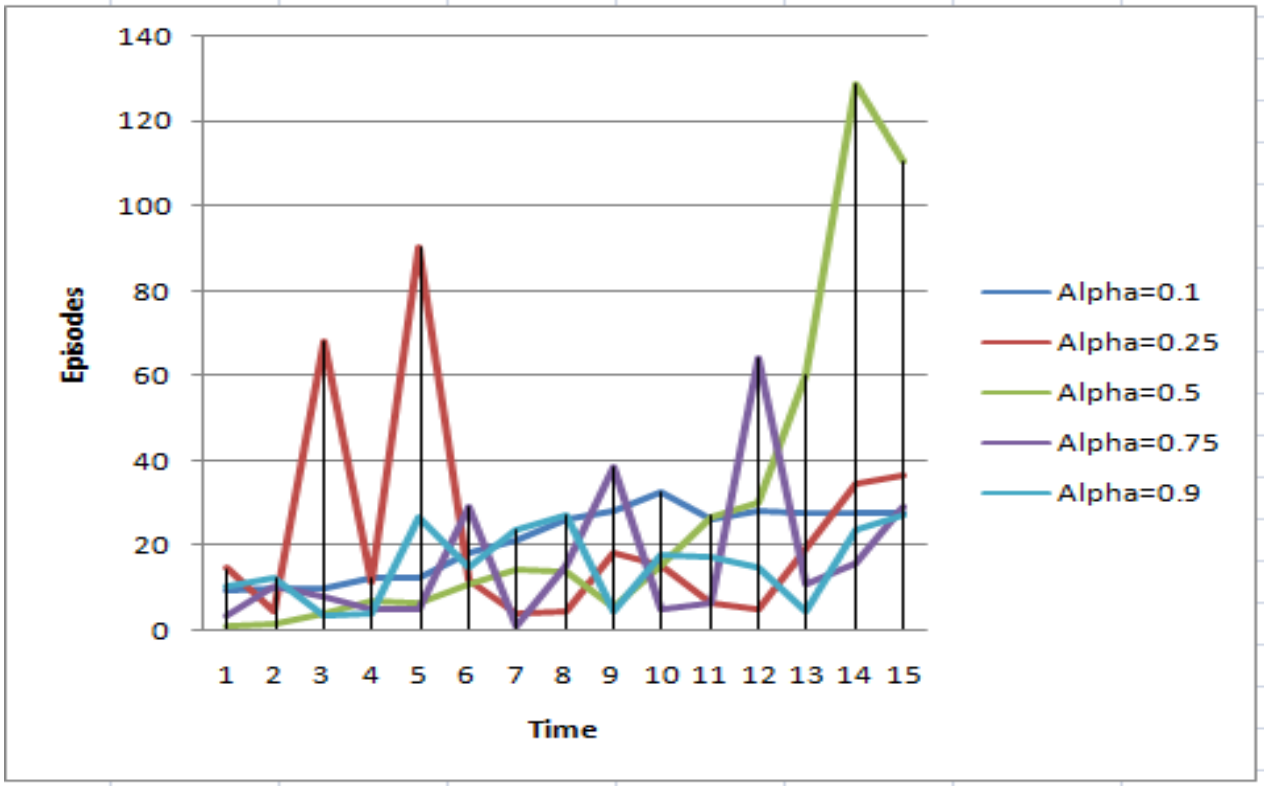


Fig 13: Different values of alpha producing chunks of reward

7.4Convergence Comparison

While implying both of the algorithms we found convergence in both cases. The values that we found and used to generate graphs for Q – learning are given below:

Q-Learning	
Episodes	Reward
1	6.85
2	7.39
3	7.60
4	7.91
5	9.99
6	10.17
7	10.65
8	10.80
9	13.61
10	15.26
11	16.30
12	17.7
13	18.04
14	18.61
15	20.32
16	20.46
17	20.40

18 19	20.50 20.15
20	20.47
21	20.46

The values that we found and used to generate graphs for SARSA lambda algorithm are given below:

SARSA Lambda Algorithm	
Episodes	Reward
1	6.18
2	4.01
3	2.59
4	4.22
5	6.75
6	6.18
7	8.85
8	8.12
9	10.76
10	11.34
11	10.35
12	18.56
13	19.21
14	22.30
15	23.45
16	30.31
17	31.32
18	30.32
19	30.04

20	30.29
21	30.26

The following figure shows us the early convergence of Q-learning.

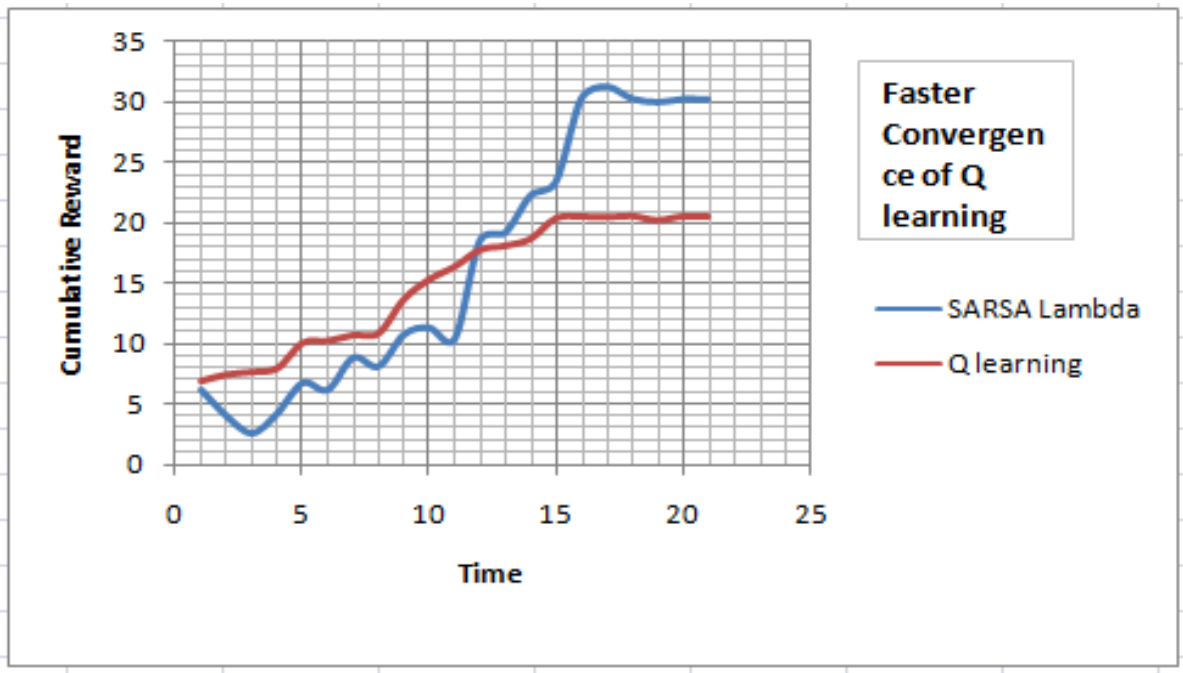


Fig 14: Early convergence of Q-learning

CHAPTER 8

8. Conclusion and Future Work

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 this 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.

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. Currently we are working on to implement our model in two of the cloud simulators “CloudSim” and “ICanCloud”. Implementing our model with 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] 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.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] P. Mell and T.Grance, "The NIST Definition of Cloud Computing." (2011): 2-3. National Institute of Standards and Technology. Web.

[9] G.Haynes. "IaaS, PaaS, SaaS, & the Cloud 101." (2014): n. pag. Web. 11 Mar. 2015. <https://www.linkedin.com/pulse/20140907071547-305726885-iaas-pass-saas-the-cloud-101>.

[10] <http://marketrealist.com/2014/07/must-know-cloud-computing-services-and-models/>

[11] K.C.Gauda, A.Patro, D.Dwivedi, and N.Bhatt. "Virtualization Approaches in Cloud Computing." 12.4 (2014): n. pag. Print.

[12] S.James and R.Nair, "The Architecture of Virtual Machines." (2005): n. pag. Web. 11 Mar. 2016.

[13] 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>.

[14] <https://eucalyptus.atlassian.net/wiki/display/DS/Dev-test%3A+large-vmware >

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

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