ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF GRADUATE STUDIES

DEPARTMENT OF COMPUTING

Cost Efficient Resource Scaling Algorithm for Cloud Computing Environment

Asebe Teka

Feb 2016

ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF GRADUATE STUDIES

DEPARTMENT OF COMPUTING

A THESIS SUBMITTED TO DEPARTMENT OF COMPUTING OF ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE DEGREE OF

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

By:
Asebe Teka

Feb 2016

Adama, Ethiopia

Declaration

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other work, and that all source of materials used for the thesis have been duly acknowledged.

Declared by:

Name: Asebe Teka

Signature: _____

Date: _____

This thesis has been submitted for examination with my approval as university advisor

Confirmed by advisor:

Name: Frezewd Lema (PhD)

Signature: _____

Date: _____

Place and date of submission: Adama, Feb, 2016

ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF GRADUATE STUDIES

DEPARTMENT OF COMPUTING

Cost Efficient Resource Scaling Algorithm for Cloud Computing Environment

By:

Asebe Teka

Name and Signature of Members of the Examination Board

FREZAWUD LEMA (PhD)	_____
Advisor	Signature
_____	_____
External examiner's name	Signature
_____	_____
Internal examiner's name	Signature
_____	_____
Chairperson's name	Signature

ACKNOWLEDGEMENTS

Though only my name appears on the cover of this thesis, there are a numbers of great people that have contributed to this work. I owe my gratitude to all those people who have made this thesis possible, because of whom my graduate experience has been one that I will cherish forever.

Foremost, I would like to express my sincere gratitude and appreciation to my advisor, Dr. Frezewd Lema for his patience, motivation, interest, immense knowledge in the subject area and the deft ways in which his lovingly challenge and support throughout the whole of this work knowing when to push and to let up. His guidance helped me in all the time of research and writing of this thesis.

I would like to thank Mr. Mohamed Kemal department head of computing; Mr. Tinsaye Gizachew for his kind help by showing direction for implementing my work; lab technician Mr. Henok that willingly helped us from giving his computer to allow us using his office as a computer lab, and many thanks to Mr. Lemessa Aschala ICT staff member cleans every obstacle regardless to network connectivity and configurations. My thesis would not have been possible without their helps.

Loving thanks to my friends and my classmates who played such important roles along the journey, as we mutually engaged in making sense of the various challenges we faced and in providing encouragement to each other at those times when it seemed impossible to continue.

Finally,I would like to express my deepest gratitude to my brother MelisTeka and sister Kelemua Teka; you are my exemplary in my whole life. You have been a source of encouragement and inspiration to me throughout my life; I specially thank you for providing a ‘writing space’ and for nurturing me through the months of writing. Throughout my life, you have actively supported me in my determination to find and realize my potential.

Most of all, thanks to God, the Divine, who continues to make the impossible to possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS AND ABBREVIATION

ABI	Application binary interface
ALS	Adaptive list schedule
AMI	Amazon machine image
AMMS	Adaptive min-min schedule
API	Application program interface
ASG	Auto scaling group
AWS	Amazon web service
CC	Cluster controller
CDN	Content delivery network
CLC	Cloud controller
CPU	Central processing unit
CRM	Customer relationship management
DB	Data base
DHCP	Dynamic host configuration protocol
DNS	Domain name service
EBS	Elastic block storage
EC2	Elastic compute device
ERP	Enterprise resource planning
FC	Fabric controller
GRE	Generic routing encapsulation
GUI	Graphics user interface
HLL	High level language
HTML	Hypertext markup language
HTTP	Hypertext transfer protocol
HTTPS	HTTP over SSL
HW	Hardware
I/O	Input out put
IaaS	Infrastructure as a service
ICT	Information communication technology
IP	Internet protocol
ISA	Instruction set architecture
iSCSI	Internet small computer system interface,
J2EE	Java2 platform enterprise edition
KVM	Kernel virtual machine
LAN	Local area network
NC	Node controller
NIC	Network interface card
NoSQL	None SQL
NTP	Network protocol

NTP	Network time protocol
OS	Operating system
OVS	OpenVswitch
PaaS	Platform as a service
PCPU	Physical CPU
PHP	Hypertext Preprocessor
PM	Physical machines
PNI	Physical network infrastructure
QCOW2	KVM image format
QOS	Quality of service
RAM	Random access memory
RDS	Rational database service
REST	Representational state transfer
RL	Reinforcement learning
RPC	Remote procedure call
S3	Amazon storage service
SaaS	Software as a service
SC	Storage controller
SDK	Software development kits
SLA	Service level agreement
SNS	Simple notification service
SOAP	Simple object access protocol
SQL	Structural query language
SSL	Secure socket layer
SW	Soft ware
URL	Uniform resource locator
UUID	Universal unique identifier
VCPU	Virtual CPU
VM	Virtual machines
VMM	Virtual machine monitor
VNC	Virtual network computing
VNI	Virtual network infrastructure
VPC	Virtual private cloud
VPN	Virtual private network
VXLAN	Virtual extensible LAN

ABSTRACT

Cloud providers have plenty of resources on their resource pool to be provided for the cloud users based on their agreement of supply from the providers' side and afford capacity from the users' side. This resource should scale up and scale down based on user demand, any cloud provider should aware of resource provisioning to be clear from resource fragmentation ,resource contention ,over provisioning and under provisioning. Most well-known cloud provider use horizontal scaling scheme for scale up resource which is take 2-5 minute to configure virtual machines, needs its own load balancer and also low resource utilization . When configuration takes place, user forced to pay extra fee for that minute. Other option is vertical scaling, resources are added on running virtual machines without shutting down the virtual machines with a virtual machines configuration time of 5 seconds which are not support on this well know cloud platform. This work therefore addressed this gap and tried to implement cost efficient resource scaling algorithm in open source cloud management software OpenStack to have an advantage of better resource utilization. We use in our experiment OpenStack platform and XEN hypervisor and we try to create 3 virtual machines .To make scale the resource we are following thresh hold value and time sires method with fixing the thresh hold value to 80% .from this experiment result we get our result is prediction error is less than 2>% and minimize monthly cost by 0.2% with a case window azure cloud provider which is better than previous related works.

Key word cloud computing, vertical scaling, horizontal scaling, virtual machines, threshold

CHAPTER ONE: INTRODUCTION

1.1. Background

These days technology plays enormous roles in making day to day human activities easier. Previously established institutions need to reconstruct their data center to run their business in an easy and comfortable manner. This would be a journey from more coasty and time taking process to cheap, time saving and effective process. More importantly, cloud computing provides computing service to the user through internet. Hence, thanks to technology, within a second any kind of information may be in your hands if you are conformable to it.

Cloud computing is a computing system where large resource pool of different computing systems are connected via network to provide different service from inside or outside that domain. These pools of resources are requested depend on service demand, the user may ask for provider scalable variable or fixed infrastructure for dump on or use from the application, different data and file storage for execution of what individual or organization. With the advent of this technology, all services are given to the user based on SLA - the document that contains each and every detail that the cloud provider and the user are agree on.

There are different cloud providers, which give their service with fixed cost within an hour for what the user use the resource on cloud by renting CPU cycle. Today, cloud computing paradigm has become one of the most interesting technologies with virtualization concept. This is a technique to create abstract layer of system resource and hides the difficulty of hardware and software working environment, typically, as virtual machines with processing unit networking connectivity, storage, and bandwidth. Simply it is computer within a computer, implemented in software .Virtualization commonly implemented with hypervisor technology. The cloud determines how those virtualized resources are delivered, allocated, and presented. In cloud, different applications and information is dump. It has many benefits over traditional data center. As cloud defined in [1] attributes for delivering computing services are:

- It uses Internet to offer elastic services which is dynamically get resource on different workload;
- It provides a massive infrastructure to support elastic services;

- The resources used for in the service have charged accordingly;
- The security and maintenance service are offered by cloud providers;
- It is cost-effective due to resource multiplexing.

The cloud enables its users to reach best resource utilization by scaling vertically and scaling horizontally according to their design. Horizontal scaling is creating a new virtual machine when there is resource shortage by copying application instance in to new virtual machines, it takes its own time to configure virtual machine and also need load balancer between virtual machines .vertical scaling is adding resource on running virtual machine when there is resource shortage on running virtual machine.

Most cloud provider like Amazon has given their service by scale horizontally their resource to users. Customers pay with CPU cycle for every request by creating virtual machine. Other providers design with scale vertically the resource concept rather than creating virtual machine for each request; they try to scale vertically the resources that benefit both the provider and users. The other mechanism is dynamic scaling either scale vertically or scale horizontally based on the load capacity. Change in capacity can be accomplished with little or no human intervention. The system scaling can happen, based on user-defined rules. Dynamic scaling provides the ability to respond nearly too changing external or internal situations and react automatically. The providers are accountable to manage this scaling issues to use their resource accordingly and also should be aware of resource contention, fragmentation, and scarcity, over and under provisioning that may lead to fail of resource allocation since customers need on time resource allocation without any failure of the resource compared with what is paid for it because cloud also can be defined as getting service as you pay.

There are three cloud delivery models:

- Software-as-a-Service
- Platform-as-a-Service and
- Infrastructure-as-a- Service

And also we have four deployment models:

- Public
- Private
- Community
- Hybrid clouds.

The aim of this thesis is to analysis different public cloud computing to able efficient resource allocation by scale vertically the available resource. Especially on OpenStack, in which open-source virtualization management software allows users to connect various technologies from different vendors and expose a unified API, regardless of the underlying technology. With OpenStack, one can manage different kinds of hypervisors, storage components, network devices and services, using a single API that creates a unified data center fabric. OpenStack is, therefore, a pluggable framework that allows sellers to write plug-ins that implement a solution using their own technology, and which allows users to integrate their knowledge of choice. Even if it is well known virtualization management software, it is not enabled to have the capability to scale vertically the resource rather than it only scales horizontally the resource; or if we need to scale vertically, the administrator must configure on dashboard manually which is not dynamic [2]. Hence, we need to explore better algorithm of vertical scaling solutions which is more advantageous than horizontal scaling that is time taking and costly.

1.2. Statement of the Problem

Cloud services are characterized by a competing interest between the cloud provider and the cloud customer. As in any business, the provider always intends to maximize profit while keeping the customers satisfied - the customers always need the best quality service with reduced cost. To achieve its goal of maximizing profit and satisfying customers at the same time, a cloud provider should have a mechanism to avail the right amount of resources at the right time to the right customer. When doing so, the cloud provider should be aware of resource contention, fragmentation, and scarcity, over provisioning and under provisioning that may lead to resource allocation failure [3].

Resource contention mainly arises when the two or more cloud enabled applications are requesting similar resources during the same period of service. Scarcity of resources arises when there are limited numbers of resources that are available in the cloud. Resource fragmentation arises whenever the cloud application request is inaccessible; in other words, there are enough resources but the service is not able to allocate to satisfy applications demand. Over provisioning is when the cloud service consumers get more resources from the service providers than the demanded resources. Finally, under provisioning occurs when the application is assigned to less number.

To address the aforementioned issues, many providers use the technique of horizontal resource scaling, which is basically adding a new virtual or physical machines to the existing infrastructure. However, horizontal scaling take more time when creating and configuring virtual or physical machine as a new, which take 2-5 minute to creating and establishment of this machines [4].This horizontal scaling is not able to use resource efficiently, whenever users request providers create a new virtual machine for the requested service without analyzing surplus resource which lead to over provisioning. Horizontal scaling also need load balancer and takes more complex. This problem is mainly for maximizing cost on cloud provider whenever the resource is allocating to customer; most cloud provider prefer to assign horizontal scaling as they cost on hourly billing systems [7] from customers. The other option to horizontal scaling (HS) is vertical scaling (VS) which is not used in many popular platforms like the open source OpenStack [5]. In this research, the researcher has worked to fill this gap by implementing VS for the OpenStack

1.3. Objective of the Study

1.3.1. General objective

The principal objective of this thesis is to propose cost efficient resource scaling algorithm for cloud computing environment.

1.3.2. Specific objective

- To review and study the works done so far in the area resource allocating and scaling
- To design cloud environment using OpenStack
- To allocate resource for cloud users
- To predict work load for virtual machines
- To create resource stack for providing information

- To implement an algorithm for vertical resource scaling
- To test and analyze the algorithm performance

1.4. Scope of the Study

This study focuses on the main problem that users make more extravagancy when they need to use resource in cloud computing environments. As stated in the Statement of the Problem section, a scientific solution has to be found to minimize cloud users fee as most cloud providers have only horizontal scaling scheme. Besides, customers can not scale their resource vertically since it has got its own effect on configuration virtual machine. The thesis therefore provides an experiment based study regarding to scaling cloud resource.

Hence, the focuses of this study is delimited to the following points:

- Providing an in-depth analysis of experimental results regarding the performance measures of newly introduced algorithm on vertical resource scaling.
- Providing resource based on cloud user request
- Creating public cloud using OpenStack cloud management software
- Checking our algorithm performance

1.5. Research Questions

The following two basic research questions have guided, and answered in this study:

1. What are the best algorithm and the issues to address in order to implement VS for popular cloud platforms?
2. How can VS help users to get best quality of service with reduced cost?

1.6. Significance of the Study

Cloud computing is a computing paradigm where a large pool of systems are connected in private or public networks to provide dynamically scalable infrastructure for application, data and file storage. With the advent of this technology, the cost of computation, application hosting content storage and delivery is reduced significantly. Cloud computing is a practical approach to experience direct cost benefits. Most providers give their service with fixed bundle instance and hourly billing system within horizontal scaling scheme. Even if some have automatic scaling

scheme, they still use horizontal scaling scheme but make with dynamically; that is why they call with automatic scaling. This mechanism of system delivery by cloud providers has its own effect on both customer and provider. As discussed below, conducting such a research therefore benefits customers and service providers significantly.

1.6.1. Significance with cloud customer

Current cloud providers use horizontal scaling at peak demand time of resource with drawback of configuration and start up time of virtual machine an average take 2 – 5 minute. Cloud users have high interference on execution their application and they pay including with configuration time. Instance paid for from user are not used fully utilized offered by cloud provider. This makes the customers to over pay without using the allocated instance. This show that currently cloud user payment for the service is not as such fair.

1.6.2. Significance with cloud providers

Horizontal scaling is poor in effective resource utilization since it creates virtual machine within every request; it does not have time to check whether there is surplus resource or not . But if they use vertical scaling, their resource utilization is high, whenever users request the providers may have surplus resource and they are not suffered by resource until they allocate. The other one issue is when creating virtual or physical machine it is complex for managing purpose; providers must use load balancer to manage all virtual machines.

1.7. Design Methodology

In designing the algorithms, different methods and tools have been used.

1.7.1. Methods

The following were the steps the researcher has gone through in doing this study:

- First, a problem identification job has been done. To do so, different literatures and documents have been reviewed.
- Second, an interview has been done with individuals working on the same discipline. These are individuals working in ICT office.
- Third, with the aim of having some more clue on the issue under investigation. A data center has been observed.

- Then, the experiment lab has been set up to deploy cloud that is accessing publicly to have solution for the defined problem; this cloud uses with open source software cloud management software which is called OpenStack.
- Once the cloud was deployed, the job of analyzing the selected solution and identifying the appropriate ones to solve the identified problem has been started.
- Finally, the solution has been implemented and tested whether it fulfills the service level agreement or not; if it is not we again propose other solution

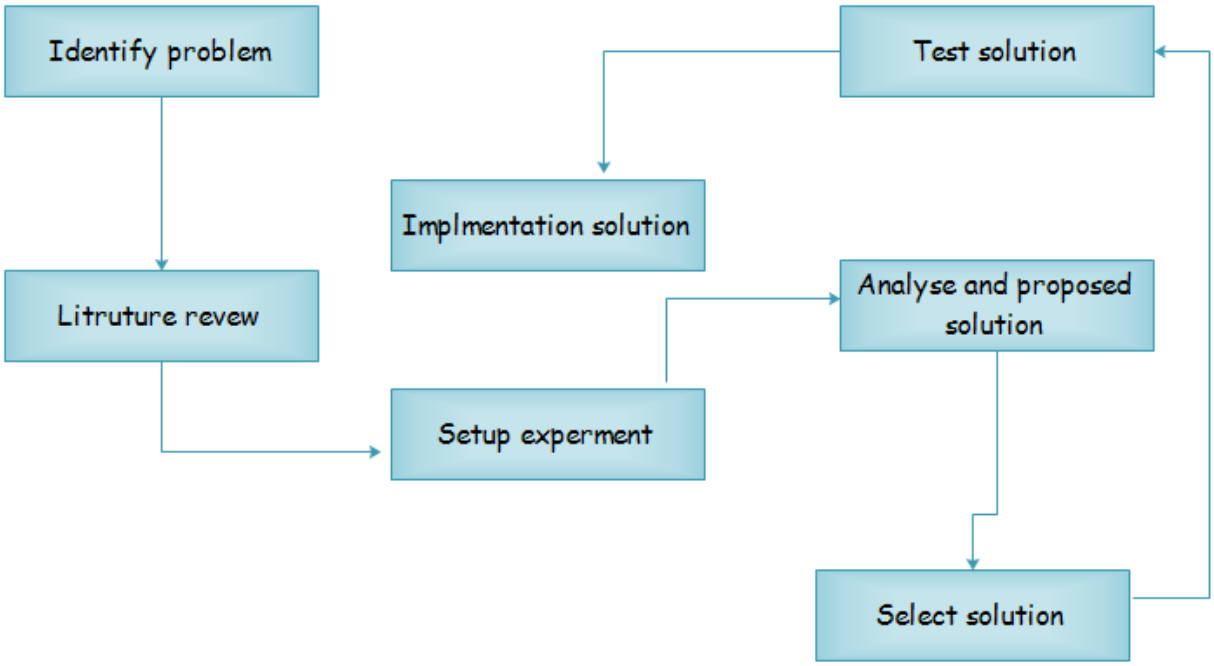

Figure 1-1:- Design methodology

1.8. Organizations of thesis

In the literature and related part, we tried to show pictorial view of different resource allocation methods through cloud applications. We tried to explain and demonstrate the majority of their features, enumerate their strengths and weakness while configuring and testing their capabilities.

In the proposed solution part we try to design our algorithm for fill the gap, how our solution is work also explain in detail and the reason why we have chosen the method we implemented is also answered.

In experiment part we try to see in detail how to set our experiment and what kind of hardware and software are used to implement our proposed solution and configurative tasks of the cloud application are discussed in detail.

In implementation and result analysis part we implement our algorithm and we try to analysis the result in detail will be discussed, what kind of result are gain from our experiment. And also how it would be work and evaluate the result by comparing with different mechanism discussed in detail in this chapter.

In conclusion and future work section we try to make conclude our work and set future work for the next researcher is stated.

CHAPTER TWO: LITERATURE REVIEW AND RELATED WORK

2.1.Literature review

2.1.1. What is cloud computing?

When the first mainframe computer is announced, it was very expensive and huge, then most researchers were looking for minimizing the hard ware part ,and they intended to make it very cheap which lets everyone to access it sitting in their own homes. McCarthy (1961) said that "for the future every computing to be like a cellphone for use of the people". Ultimately, the hard ware became as what has been said.

Todays the most cloud Provider Company such as amazon, Google, eBay, etc. are investing a huge amount of infrastructure to set up their data center. When they allow their data center to customer, they had to maintain the servers to sustain their peak workload resources. This turns into a viable economic model to rent the resources to public. Amazon launched a service called Amazon Web Services (AWS) [7]for serving cloud.

There are different definition about cloud computing. Among those defined by NIST [8] is one as it is a model for enabling on-demand access to a share computing resource that rapidly allocates and de-allocates with interaction of service provider. Whereas, in [9] ,cloud is stated as it is a type of parallel and distributed system consisting of virtualized computer which is connected each other and dynamically allocate and de allocate computing resource based on SLA as established between provider and customer

On the other side, in [10] , it is stated as a combination of powerful computing resource that is used to accelerate new era of IT for provider and customer service which is available on demand based on customer's economic capacity

In [11] , is stated as the next stage in the Internet's evolution, serving everything from computing to business whenever you need. In line to this, Cloud by itself it is not a new technology, but it is combination of different existing computing having the concept of virtualization which consists of clusters, grids ,virtualization, Internet and service oriented architecture. Clusters are distributed and parallel systems that are ruled under single administrative domain. The node in the cluster integrates to form a single computing resource [12].

Grid is aggregation of autonomous resources that are geographically distributed. In grid computing node at run time allow selects and shares dynamically [12]. Cloud' is an elastic execution environment of resources touched with many stakeholders for specified QOS or SLA. In cloud computing you can accessed the resource at any time and from anywhere , where resource is requested or released through web interface or well defined API for scaling resource .This characteristics is able to on demand access that whenever the application needs them and can be released when un used . The services offered by cloud for requesting and releasing resources are done in dynamic manner, that is, without human intervention.

Generally cloud computing has different deployment and delivery model, characteristics, stockholders, that is depict in Fig 2-1.

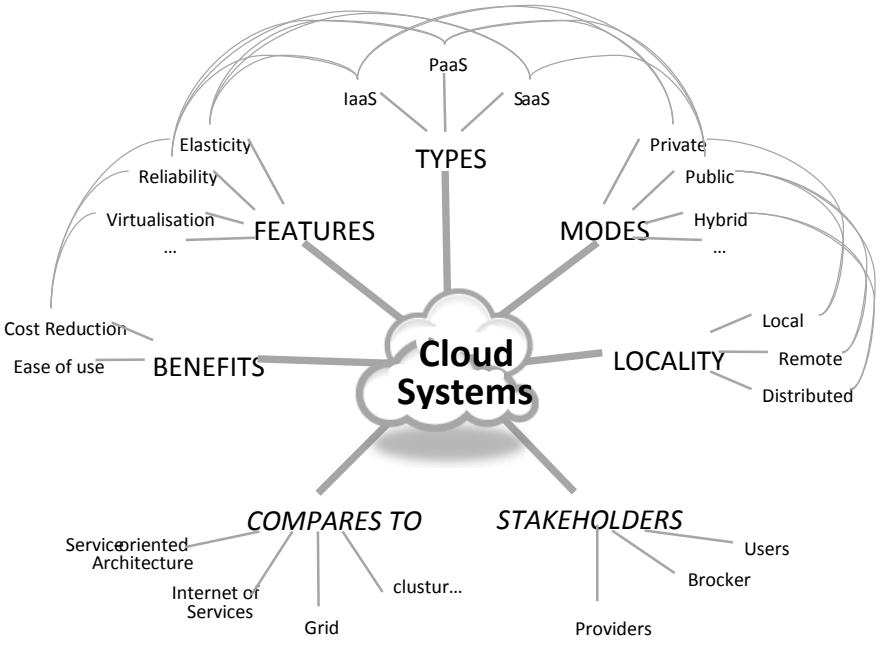

Figure 2-1:- non-exhaustive view on the main aspects forming a cloud system [13]

2.1.1.1.Types of clouds delivery model

Cloud provider have full of resource that expect their resource pool out for the cloud user based on their agreement recognizing their supplying from the provider side and afford capacity from the user side. Typically the providers rent with different cloud functionality, Infrastructure, Platform or Software, Cloud services can be classified based on the level of abstraction they propose. There are three commonly accepted layers of abstraction in cloud computing, the services from the upper layers usually rely on the layers underneath

2.1.1.1.1. Infrastructure as a Service (IaaS)

The IaaS layer represents the lowest level of abstraction in the cloud which basically provides enhanced virtualization. Providers serve resource with different service interface .Cloud supplier provides an online infrastructure for which customer can store data, develop and run any applications as their interest. Cloud Providers offer the capability to provide computing resources which are typically virtualized. To execute cloud services and applications, Cloud users are provided with virtual machine instances that have different or similar operating system running with the host machine that installed different software with requested resources in terms of CPU frequency as well as number of virtual CPUs, memory, bandwidth and storage capacity. The IaaS provider also enables the users to manage the networking of their VM instances either by restricting their access or setting up virtual networks. Furthermore, users are responsible for middleware, runtime, data, and applications levels. Basically users rent a virtual machine (VM) with their preferred OS installed. The provider generally does not care what users do with this VM. The users can as well create snapshots of their instances as a means of backup, or use the snapshot of a running VM instance to create a VM template which can be later used to provision new clone instances.

A fundamental building block of a cloud computing infrastructure is a server. Cloud computing servers are used to deploy VMs on which applications can be run. A cloud provider also provides various forms of data storage. Users are given privileges to perform certain activities on the server [14], such as starting and stopping a VM, and configuring access permission(Amazon EC2, Zimory, Elastichosts).Amazon has been a pioneer of IaaS with its Elastic Compute Cloud (EC2) [15]. Nowadays, it provides three pricing models for its VM instances. These are

- Users can classically pay for their on-demand instances on an hourly basis,
- Via a prepaid fee, users can pay less for reserved instances that they plan on using heavily
- Users can bid the price they are willing to pay for a spot instance and as long as the price of these instances is computed over the different bids of the various users, is less than the one they specified which is with provisioned instances .While Cloud users are charged for out-going traffic and storage related to the instances disks' but Other IaaS providers might charge usage per minute instead of Amazon's hourly increments.

2.1.1.1.2. Platform as a Service (PaaS)

In PaaS, the cloud providers have given any kind of platform that able to the user to develop any kind of software or applications .The platform could be programming languages, commonly used libraries, graphical user interfaces, software testing environment, operating system, so on. It makes easier for developer to develop and deploy a new or customized software on cloud infrastructure. Software developer focus on only on developing their application rather they worry about the maintenances and upgrading of the platform they used which leaves for PaaS provider. However, some PaaS providers ask the customers to configure some parameters that help maintaining the performance. For example, Google app engine [16] asks the customer to select the front-end instance class from three different classes of instances of which each class has different CPU frequency and Memory size. Moreover, it enables the customer to predict the number of required instances. This number helps the provider to prepare VM images in advance and avoid the delay of copying the code when a new instance is needed. To mentions some example, providing cloud with PaaS, are Google App Engine enables hosting and developing web applications in Google-managed data centers, programing tools [17], Amazon's Elastic Beanstalk [18] provides sandbox capabilities on Amazon's infrastructure and Microsoft's Azure [19] provides enterprise database services by way of APIs, software development kits (SDKs) for programming languages like .NET, Java, PHP, Python, and many other services.

2.1.1.1.3. Software as a Service (SaaS)

Providers offer specific business functions with specific cloud capabilities, i.e. they provide applications (services) using a cloud infrastructure or platform. SaaS cloud providers offer ready-to-use cloud applications for their consumers who access its features via intranet or the Internet.

SaaS is mainly give freedom to cloud user that they need only to have an interface to communicate with internet maybe Smartphone, personal computer or any device that can accesses internet or intranet. It allows clients to use business software services, such as accounting, CRM, ERP and several other types for a monthly fee instead of having to design, develop and host it on their own. The users are then completely freed of the development and maintenance burdens. Because of it is a software delivery method that provides access to software and its functions, hosted on a cloud it is highest level of abstraction. SaaS applications are designed using latest web technologies that are well suited to run on Internet. SaaS providers can deliver services using two architecture model (multitenant and sing tenant model).

The multi-tenant architectures that multiple customers use the same software applications with the same functionality and configuration capability , the data is stored with the shared database with its drawback on QOS, and the other hand , single tenant ,that has its own separated environment for the software applications is more secure than multitenant . In this architecture, the user can configure the software as their need. Besides, it is expensive. Based on customers' Business demand, architecture types must be selected on the following selection criteria: configurability, customization, and security. The most well-known SaaS applications which are being used by everyday life include Google's mail service and Documents application. Though those application are accessible to the general public, organizations may subscribe to them for internal use

2.1.1.2.What are cloud deployment types

Clouds can be hosted in different fashions, depending on the business model what the provider or the user need to be used. The tendency of clouds provider to evolve in the market begin from internal solutions which include the management of the local infrastructure and the amount of requested data to external solution which enable to access the entire world. Currently there are different providers have gained with external market. There are different types of deployment method to access the market public, private, hybrid, and community.

2.1.1.2.1. Public Clouds

Public cloud usually consist in huge interconnected data centers managed by a cloud provider and it has not restriction on who use the cloud any one can use this cloud on a charge-per-use basis.

Hear in public cloud any computing resource are rent for user instead of buy and maintain, that is any computing power are scale in and scale out in the datacenter only different organization may use cloud functionality from others and also they can offer their service to users which are outside the company .Providing resource for user with the actual power for its own purposes allowing to minimize users cost and effort to create their own infrastructure. Public cloud provider grant their different infrastructures to users are via internet SLA must be signed between provider and users certain level of QoS in the services provided. The security and privacy of users' data is managed by the service provider, as the data is hosted on provider's cloud .Some customers are reluctant to choose a public cloud due to privacy, policy, and security concerns when their application operates on sensitive data .Generally Public clouds provide an elastic, cost effective means to deploy solutions for users Examples of public cloud services include: Microsoft's Windows Azure Platform, Amazon's AWS, and Google's AppEngine

2.1.1.2.2. Private Clouds

Private cloud are an internal data center of business organization that provider offer service for a specific group or organization and access is limited to that group or organization which is pay for the service get from provider . It is privately managed by an organization for its own use. The organization get benefit from cloud infrastructures' flexibility and user-friendliness while guaranteeing the privacy of sensitive data .Except some futures, all functionality are not directly exposed to the customer, i.e, it owned by the customer but frequently built ,installed, and managed by third party which is by provider .

A private cloud is less cost effective than a public cloud, but may be more suitable when an application must process sensitive data. Public cloud provider can serve specific resource within their public cloud infrastructure to build virtual private cloud , that customer can be assured their data is stored on and processing only on dedicated servers and these servers are not shared with any other customer of the cloud provider. A private cloud take an advantage over public cloud, such as being elastic and service based. The difference between a private and public cloud is that, in a private cloud-based service, the organization managed and process the data with full privileged of network bandwidth, security exposures and legal requirements. There are different cloud

management software able to have private cloud like OpenStack and eucalyptus. We can also list as an example that give private clouds are eBay and Amazon.

2.1.1.2.3. Hybrid Clouds

Organization need some data to be more confidential but they also not to be confined their service within limited customer to outsource infrastructure to cloud providers, they at the same time would lose control over the resources and the management of code and data out for customer refers as hybrid cloud, different organization might utilize this approach because they want to exploit the scalability and cost-competitiveness capabilities that a public cloud provider offers, but they also want to keep their sensitive data on their own premises or in a private cloud e.g. sensitive data by employing local private clouds. It is multiple cloud systems that are connected in such a way that programs and data can be moved easily from one cloud to another cloud. Hybrid clouds can also be put in place as a result of cloud bursting: Cloud Bursting. Cloud bursting refers to a service deployment model in which application runs in a private cloud and gusts into public cloud when the workload increases to an extent such that private cloud cannot handle it. E.g. cloud bursting for load balancing between clouds [20]. Some examples are providing hybrid cloud is IBM and junipers are listed.

2.1.1.2.4. Community Clouds

Different organizations share the same infrastructures that have similar activity to make their activities more reliable for its customer which have common attributes (e.g., mission, security requirements, and policy and compliance considerations). The cloud is called community cloud. For example, different bank can federate their private clouds to have one community cloud that is available to every bank in the community witch the customer from one bank can use from other bank which is in part of the same community. The cloud can be managed and operated by one or more of the organizations in the community or another third party. Community clouds can either aggregate public clouds or dedicated resource infrastructures.

2.1.1.3.Characteristic of cloud computing

This section specifies different characteristics that mainly express in cloud computing environment, which the user may face.

2.1.1.3.1. Elasticity and scalability

Elasticity the ability of underlying infrastructure to adapt to changing, it enables users to acquire and relinquish the computing resources dynamically. Cloud provider facilitates any kind of resource usage based on the request hence users need not plan for futuristic usage. The computing resource allocations can increase or decrease according to the consumer's demand. This change in resources is called elasticity. Elasticity enables scalability. In cloud there are deferent types of scalability, horizontal scalability which create virtual machine for every request, and vertical scalability which scale in the computing power within virtual machine. Elasticity provides a convenient way for the scalable applications to scale resources .For example amount and size of data supported by an application, number of concurrent users etc.

2.1.1.3.2. Reliability

Is denotes the capability to ensure constant operation of the system without loss of data. Reliability is achieved through redundant resource utilization. Cloud providers usually have more than one data center and further reliability can be achieved by backing data up in different locations .particularly it focuses in prevention of loss of data. It is already built in many cloud solutions via storage redundancy. e.g., in Amazon reliability concern once-a-week failure rate

2.1.1.3.3. Multi tenancy and Resource pooling

Resource pool is a collection of different resource that are not assigned to the user but assigned accordingly user request. Different physical and virtual resources are pooled then assigned to specific consumers for their use according to their customers' demands .Multiple customers can be allocated resources from same physical server, which is called multi tenancy. It improves the overall system utilization of the infrastructural resources on account of resource sharing. Typically cloud providers, allocate their resources in order to serve multiple customer using a multi-tenant model, which. the resources are generally location independent, thus the customer generally does not know the location of resources, however, it is possible in many cases for the consumer to specify the location at a higher level of abstraction (e.g., country, state, or data center) .When the same resource are assigned to multiple users at the same time it is obvious that it affect infrastructure resources as well as data services that are hosted on shared resources. Naturally, all information is saved in separate databases, but information is concurrently altered in more complicated way, even though maintained for isolated tenants. Resource pooling have different

advantage among that the provider does not permanently assign a particular resources to a specific individual customer until there is request (i.e. the available resource is efficiently utilized for other request, but when the granted user is request the resource dynamically assigns resources).the other advantage of resource polling are facilitate increasing reliability and user can add and remove resource.

2.1.1.3.4. Pay per use

This property shows that fairness of that the customer pay for the resource that he/she consume only. The cloud workloads are classified into two groups: time critical applications and applications with soft time requirements [21]. First group consist of web applications that have fast response time are critical for the whole operation, if the response time is late the provider may lose their customer eg. Online book store Second group consists of applications running back-end tasks .This group of applications do not require real time responsiveness eg, Hadoop. Users which have time critical application can request more resources for a short time interval to complete the job. This pay per characteristics strongly relate with quality of service support, where specific requirements to be met by the system and hence to paid for can be specified.

2.1.1.3.5. On-demand Provisioning

Resource can be request or release, based customer needs, this provisioning should done automatically granted or take away to/from user. To perform tasks such as building, deploying, managing, and scheduling, a cloud computing environment should allow the user to interact with the cloud in such a way as to be able to explicit reserve and return resources .But, this may depend on paying ability of the user providers service is not free service. Is perform based on user request which the user can pay for the resource, i.e., the time they were in possession of the resources. The reactivity of a cloud solution, with regard to resource provisioning, is indeed of prime importance as it is closely related to the cloud's pay-as-you-go business model.

2.1.1.3.6. Universal Access

Cloud services are available over the network through high speed internet. Clients from anywhere and time can access the cloud via Internet This enables the users to access their cloud resources using any type of devices, provided they have an Internet connection .Provisioning resources in the cloud to host applications implies accessing these applications through the Internet by a vast

number of clients. To offer a high QoS for all clients, cloud providers should have standardized network connections that have large enough capacity coping with the increase in the number of hosted applications and clients.

2.1.1.4.Cloud Computing Benefits

When enterprise facilitates their business with cloud computing technology they get different advantage that can run their business smoothly. Among the most but not the list are listed as follow

2.1.1.4.1. Cost reduction

There are different reason that the enterprise gate this advantage .when one enterprise try to establish their business, it must include with information technology .especially for beginner enterprise it may be headache to full fill this technology instead it can rent those different infrastructure ,software and platform from cloud provider that may be construct its datacenter virtually according to the business plan .to create data center in that enterprise it may take long time as well it invest more money ,instead of running its business it may cost for this data center formation if it is necessary . Administration cost also can be reduced while the cloud provider is responsible for managing, patching, and upgrading both the software and hardware systems.

2.1.1.4.2. Flexibility

Enterprise business is may change rapidly from the initial period with dramatically. Whenever change occurs in business, enterprise owner shouldn't wary about scalability issue all are concern to provider's side. Cloud providers offers large Infrastructure like storage disk. If sudden workload spikes, it can managed effectively, since resource should scale dynamically. This makes users business can achieve the goal without intention of flexibility with low cost

2.1.1.4.3. Customization

In cloud computing most users used different customized plate form to meet their interest based on latest library, platform with customization

2.1.1.4.4. Disaster recover

The very big and interesting thing in clouds are you can have disaster recovery scheme .Public clouds provider are developed for serve the user globally that any user can run his business

through the globe ,to full fill this activity data center are distributed in different location they can distributed geographically to have more reliable data via virtualization .this virtualization techniques allow to make data recovery .E.g. amazon cloud provider have different data center disturbed over glob on each datacenter have the same replica of data.

2.1.1.5.Cloud Computing Issues

While using this big technology as it has own strong side, there have also some critical issue regarding privacy, security and so on... This issues are listed as follow

2.1.1.5.1. Privacy and security

Cloud providers must keep their customer privacy since they are process valuable data from different customers on server. This data must keep that's allowed to be accessed by customer only even not by the provider also under any circumstances. They can be used different security method to protect customer privacy even if you have deal with on SLA you have not guaranty that can be robed by some on else and also even customer may not know from which data center they get service since it geographically distributed at all there is not guaranty because this is cloud.

2.1.1.5.2. Data migration

Once customer have service level agreement based on that the customers are not satisfy what the service get from the provider customer may have the right to change other provider. But hear, the great challenge is the two cloud providers platform may not be compatible the service once have May not get on the other provider and may add also more fee to get the service. So whenever customer chooses they have to check what they really need and which provider may serve their interest because at the last they may suffer to pay unexpected fee.

2.1.2. Overview of cloud platforms

Cloud computing have a collection of resource that serve service through internet .cloud users get vast resource that they are pay for resource used only .Different cloud computing providers have emerged for providing various services to customers . there are different cloud providers that based for different deployment model as well as for different service providing for cloud users such as amazon, window azure, Rackspace giving for public cloud infrastructure as a service (IaaS) which are commercialized to access this platform, there are also open source cloud platform like

OpenStack, eucalyptus, Open nebula that you can customized as provider need. Due to the diversity of the features providers, it becomes very trouble for fresh cloud users platform based on their requirements have to consider different factor as mentioned below [22]

- How is a service outage defined?
- How the customer compensated for an outage?
- Is there reporting mechanism if there is accident?
- Are access /us age reports available?
- Is the data backed up, if so, where it will be stored?
- How safe is the data?
- What is the billing model?
- Are charges based upon traffic, us age or storage limits?

2.1.2.1.Amazon Web Services

The Amazon Web Services cloud is the largest online retailer in the world, which supports its operations through data centers. For processing transaction, it require large amount of infrastructure that provides reliability and scalability with minimum cost. Amazon has well organized data center infrastructure with supporting virtualizing technology. More of investment in amazon is for data center by renting this platform and storage services to developers for developing and hosting applications. Amazon's cloud provides different service with different functionality [23]. Which are elastic Compute Cloud, SimpleDB, Simple Storage Service, CloudFront, Simple Queue Service and Elastic MapReduce and so on below the diagram depict different service that is available in amazon cloud

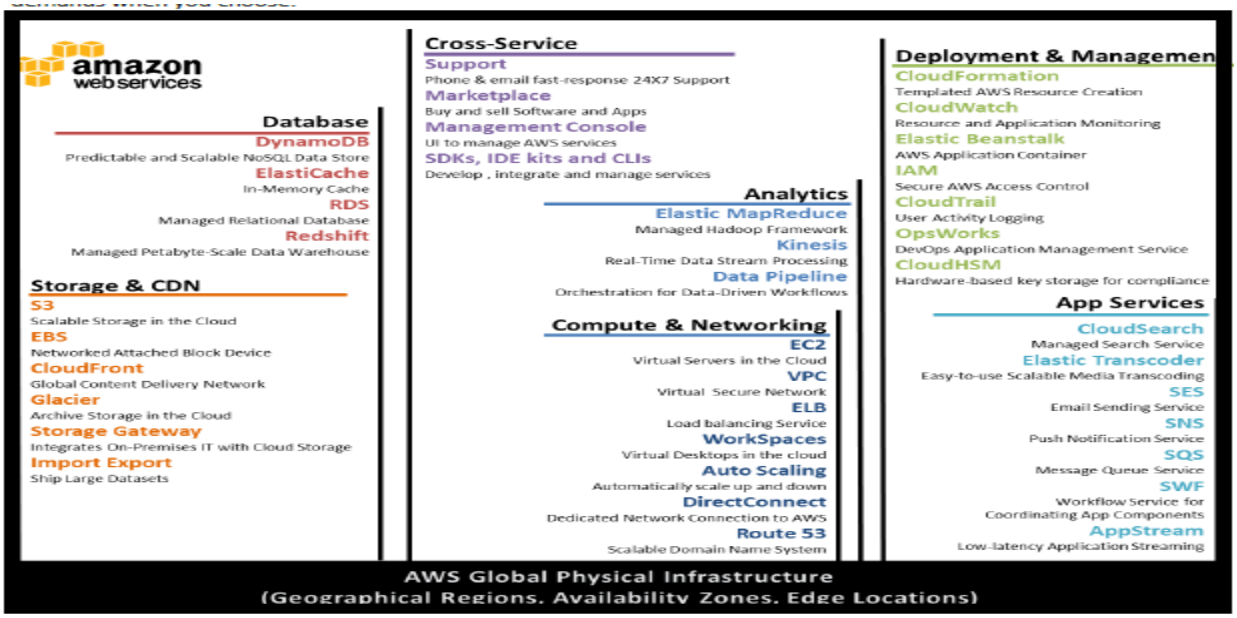

Figure 2-2:- Amazon web service [11]

- AmazonElastic Compute Cloud provides basic computation service in AWS, which provides a highly flexible and scalable computing cloud for its consumers. It has a well-documented web service API coming with available configuration and administrative tools that enable a straightforward and dynamic cloud management. It presents a virtual computing environment and enables resizable compute capacity horizontally. Users can simply use a pre-configured AMI. Every pre-configured application software and operating system creates their own AMIs [24]. Amazon offers a pay per-use pricing model based on the computing hours of running instances [25]. Pricing is per instance-hour consumed for each instance type. Each partial instance-hour consumed will be billed as a full hour. [26]. The owners of the instances get full control over the virtual machines. User can launching and termination the instant for that he paid. Instance current state, performance and resource usage can be monitored with CloudWatch service, in order to gain visibility into resource utilization, operational performance. You can create *ASG*to automatically scale your capacity based on different metrics. Amazon CloudWatch collects and displays values of built-in or even user custom performance counters. Statistics and various graphs can be generated from the metrics. CloudWatch enables to create alarms based on instance metrics. Alarms can send notifications via Amazon's Simple Notification Service (SNS), which provides an efficient and flexible publish subscribe messaging model for sending

notifications from the cloud. EC2 also offers Auto Scaling and load balancing services. Auto Scaling allows users to scale up/down their EC2 capacity automatically. When pre-define events are triggered, load balancing distributes incoming traffic across multiple Amazon EC2 instances automatically. EC2 instances can be launched in multiple locations

- Amazon Elastic Block Storage *provides* network-attached persistent storage to amazon EC2 instances. EBS volumes are create and stored on amazon S3 [23]. S3 is a storage system for EBS instance [27]. For the case of their manipulation S3 offers standard REST and SOAP interfaces. When you store your data should be protected from un-authored user, hence S3 use different encryption mechanisms (SSL) on client-side and server-side before or during uploading to the cloud and provide automatic decryption upon data retrieval. And also enables customer to implement further protection [26]

- Amazon hosts a CDN service as well, called CloudFront, which provides an easy and efficient way to store and deliver static or streaming data. CloudFront is integrated with other AWS services; it webs the whole world to enable geography-aware, high bandwidth content delivery.

- Amazon RDSprovide to operate, scale and setup a database in the cloud. You can use any database platform to launches launch a *DB Instance and get administration automatically like to take back up.*

- AWS includes several services regarding networking Route S3. The VPC service allows users to create a private, isolated network topology within the Amazon cloud and gives full control of network configuration including IP addresses, public and private subnets or routing. Placing databases or background applications in a separate backend virtual network that is not accessible over the Internet, further increases security. VPC has the ability to integrate with existing on-premises Virtual Private Networks, therefore it is possible for a company to use the cloud as an extension of their local infrastructure.

- Amazon's Elastic IP Address is similar to static IP address in traditional data centers, with one key difference. Without network admin and waiting DNS user can map an elastic IP address to VM instance in sense of account but not VM

- CloudFormation s**ervice** enables developers, system administrators to describe, then launch complete hardware, and software stacks along with all the related resources .Amazon offers several other services, which go beyond the scope of this work, as depict in Fig 2-2

2.1.2.2.Microsoft Azure

Microsoft started its initiative in cloud computing with release of windows azure in 2008 [28], which is initially for PaaS purpose that support different programing language. At these days the company owns products that cover all type of service model. Which offer IaaS and PaaS [26] Pricing is based on Compute, Storage, Storage transactions and data transfers .per instance-hour consumed for each instance type, from the time an instance is launched until it is terminated. Every instance is earn with full hour bill system [26] .Windows Azure in turn is made up of different components, namely, compute, storage, connect, fabric controller, and CDN.

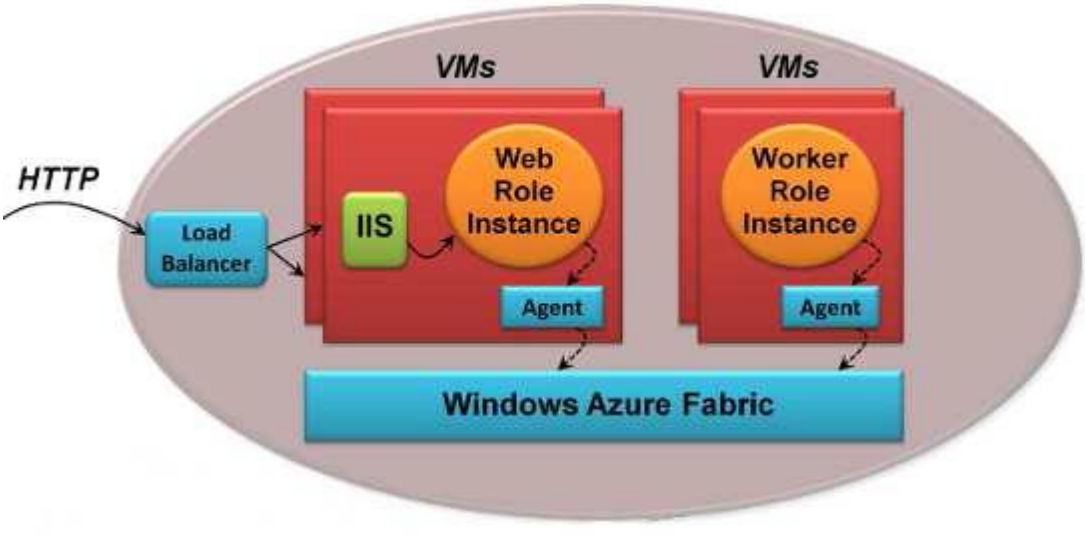

Figure 2-3:- Windows Azure architecture [29]

- With two role window azure applications are run on compute platform [30] which is a separable logical part of an application that has a well-defined task. This is web role and worker role. Web roles are used for web based applications and are invoked by web requests. They are hosted in Microsoft's Internet Information Services (IIS). Worker roles are intended to run background jobs and do not interact with the user directly. The compute component also contains a load balancer distributing jobs between the available instances. In order for the application to be scalable, which is a big part of the incentive for creating an Azure application, the instances has to be stateless since there is no mechanism to ensure a client is handled by the same instance over many requests. Client specific information has to be stored in Azure storage or by other means be made available to all of the instances.

- An application run on window azure can be on one or more VM. VM has local storage, which an application is free to use. The storage component offers three kinds of storage: blob, table and queue. Blobs are ideal for storing large, unstructured binary data. They can store large data objects, up to a terabyte, and contain metadata about the data object. To optimize transmission of these potentially large blobs, they can be divided into blocks allowing retransmission of separate blocks in case of a failure. Tables provide a NoSQL data storage solution for storing and fast accessing sizable amounts of typed data organized into key-identified groups' .these are not relational tables, and consequently SQL is not used to access the data. Instead the data is accessed through a REST API and the data objects, or entities, has properties of types as int, string, and bool. The use of a NoSQL storage model like this enables the data storage to scale and the data to be partitioned over several servers.

- The third part of storage, is queue, structure that provides a natural way for different roles to communicate. A web role that receives a request which demands heavy computations can hand it on to a worker role through a queue and the worker role can then hand it back through a different queue when the work is complete. The data stored in Azure storage is replicated three times to mitigate any data loss.

- The CDN, or content delivery network, is aimed at improving performance for data that is accessed frequently from places all around the world. By storing local copies at different

sites the access speed can be improved. The connect component enables an Azure application to connect to another computer at the IP level rather than HTTP, HTTPS and TCP which are the normal protocols for connection to computers outside of the cloud for applications on Azure. This can be useful if an application is to connect to a database on a local machine running SQL Server .SQL Azure is the cloud version of the traditional SQL Server relational database management system [31]bundled with such features as transaction management, consistent data access concurrency, etc. it is non-relational because of the ability to scale more easily. A SQL Azure database can be accessed through many well-known data management interfaces and administered by the conventional tools with either a protocol called Tabular Data Stream, the same protocol used to connect to a local SQL Server database, or Open Data Protocol. The cloud version of the SQL Server Reporting Services is called SQL Azure Reporting, which allows creating in-depth business analytics reports on the stored data. SQL Azure Data Sync allows a SQL Azure database to be synchronized with another database, a local SQL Server database or another SQL Azure database .SQL Azure automatically takes care of database replication to increase database reliability and data consistency.

- Azure's Fabric Controller (FC) pulls the strings by transparently managing the virtualized resources and VMs of role instances. Upon starting the roles, the FC automatically launches the configured number and type of virtual machines as instances, and starts running the given role's executable. When role is successfully initiated, and begins running, the FC starts monitoring the role instances. If an instance suffers any kind of software failure due to an application exception or operating system error, or hardware failure in the cloud environment, the FC immediately replaces the unhealthy instance with a new one to maintain the configured capacity The Fabric Controller is also responsible for installing patches and upgrades of the operating system and other software, such as the .NET framework or the IIS server on the instance virtual machines. Some of these upgrading operations require rebooting the virtual machine. The FC controls the application update process, it goes through the update domains one by one, stops the instances running in that domain, deploys the new code version, then starts new instances based on the updated role. By proceeding so, the FC guarantees that a situation when no instances of a role are running cannot come up. There is cache memory which is offered by Azure for storing most

frequently accessed data auto scaling Application Block (WASABi) allows defining how Azure applications should respond to varying load by horizontal scaling. WASABi is designed to encapsulate the logic for automatic scaling, so it requires minimal changes to the scaled application

2.1.2.3.OpenStack

OpenStack is a cloud management opens source software for controls large computing resource pool in data center through dash board a dashboard [32]. OpenStack offers *IaaS* solution for cloud users. Each service has an API that facilitates the integration.

- Nova :- is a compute service for managing VMM and for creating instance .the VMM is plugin to computing service nova with API

- Neutron :-is a network service for creating network service and connectivity

- Cinder :- it is storage service for managing and creating external storage including NFS and block device.

- Keystone :- it is an authentication service for users and service
- Glance :- is image service for uploading and managing images used by uses ,it is not storage , rather it save images attribute

- Horizon :- it is portal system that is a dash board to create GUI Dashboard, creates a GUI for users to manage OpenStack deployment

- Swift: - it is a service to retrieve and store unstructured data with *RESTful*, HTTP based API. Data is replica to multiple drive across server cluster. it is highly scale out and replica file

- Ceilometr:- it is service for meters and monitors the cloud billing ,scalability, statics purpose

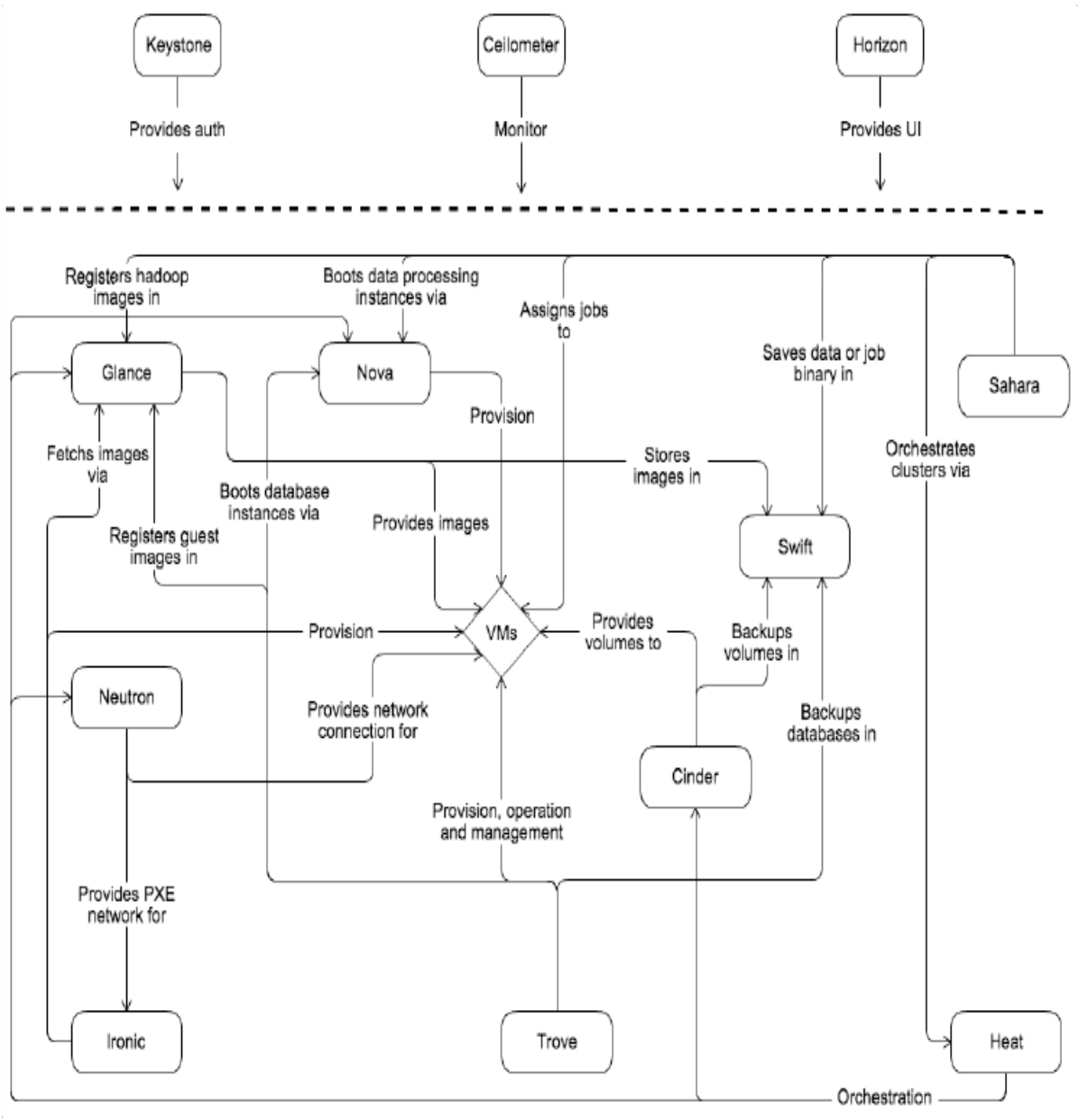

Figure 2-4:- conceptual architecture of OpenStack [33]

OpenStack only supports horizontal scaling [5] in order to match the cloud paradigm of "cloud-based applications that typically request more, discrete hardware". Consequently, the ideal for OpenStack is to add a VM and to load balance among the existing VMs in case of increasing resource demand [5].OpenStack model is a lot of related service that other open source software

does not, this make OpenStack more secure because of having many related service the security is more strong than the other open source cloud software as depict in Fig 2-5 below

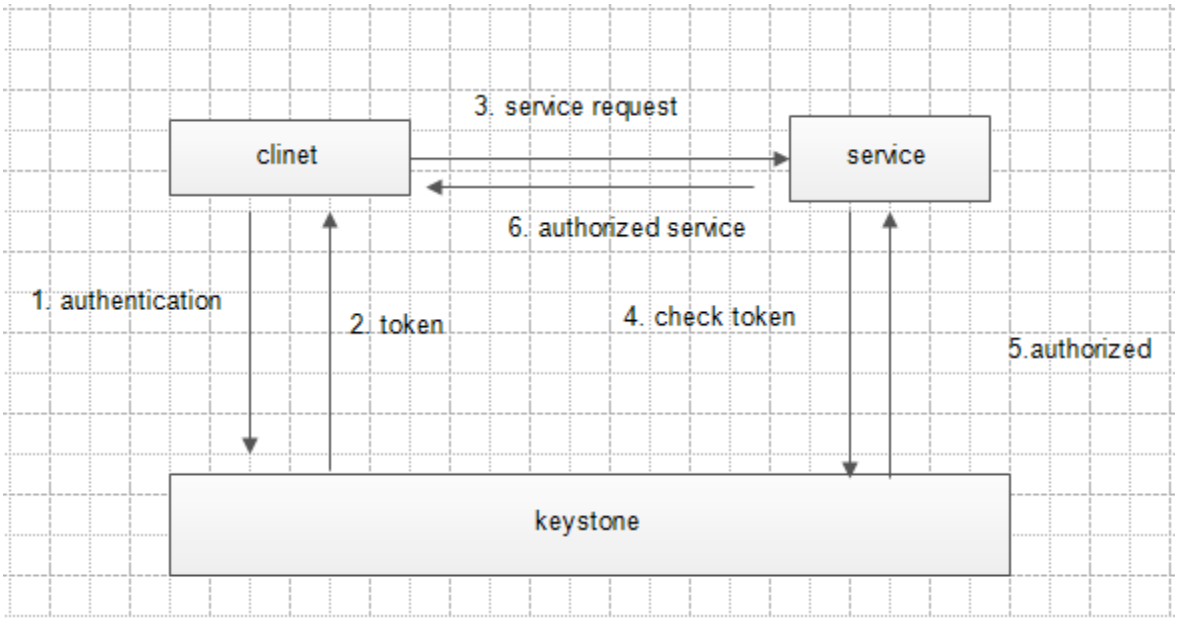

Figure 2-5:-Keystone security chart

The other issue with OpenStack is high availability with storage that has no with other open source cloud management software every object are placed with three disc for reliability purpose , if ones disk goes down with disaster or power fails data will not lose because of replica the object with different disk

2.1.2.4.Eucalyptus

Eucalyptus is Linux-based software architecture that implements scalable, efficiency-enhancing. It provides hybrids and private cloud. Eucalyptus provides IaaS and also it gives CLI called Euca2ools for interacting with web services. [34] The Eucalyptus cloud system has five high-level components—*Cloud Controller, Walrus, Cluster Controller, Storage Controller* and *Node Controller* [35]

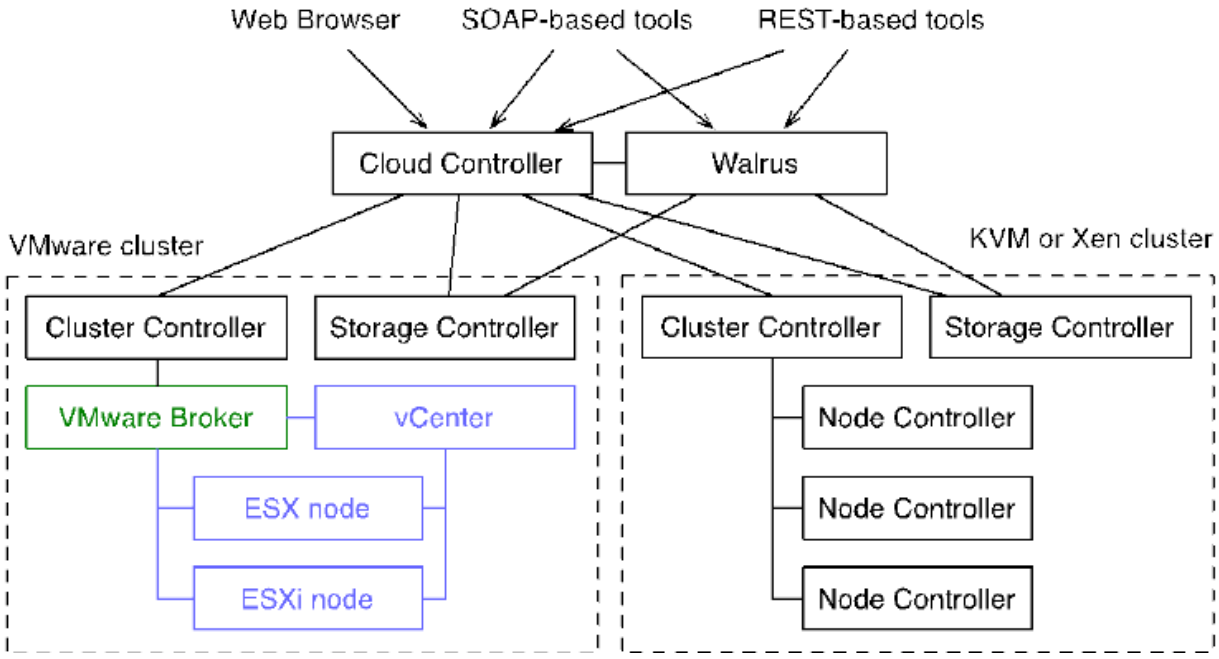

Figure 2-6:-Eucalyptus Cloud architecture [36]

- Cloud controller is an entry in cloud for users, developer, administrators and so on. The CLC is used for manages to know about resource information's and managing the verbalized resource
- Cluster Controller Generally executes on a cluster front-end machine, or any machine that has network connectivity with nod running NC and CLC. Based on information gather it set VM and VM schedules which is execute on specific NC.

- Node Controller is executed on each designed to host VM instance .it also mange inspection ,execution and termination of VM instance

- Storage Controller Ii is like EBS and have capable of integrating with different storage system like NFS, iSCSI, etc. Linux block device is act like elastic block store to attaché VM with send disk traffic across attached LAN to storage location .EBS allows to create and store in a central storage system like walrus but not shared across instance.
- Walrus provide users to store persistence data .it allow users to create, delete, put, and get objects. its interfaces compatible with amazon S3 and support AMI image interface which is provide a mechanism for accessing VM images and users data

2.1.3. Virtualization In cloud

Virtualization is a technology which is different VM is share the same computing hard ware machines. VM is mainly enhance resource sharing and utilization with application flexibility .hard ware and software resources are virtualized with different functional layers .this idea is to separate HW and SW for better efficiency . For example, paging in virtual memory was implemented in Manchester university 1959 [37]. Similarly this all resource are (CPU, memory, storage, network channel, etc.) are available within servers through distributed or n single data center.

Server may be normal or virtualized using cloud technology. On server there are different load balancing activities when client integrated with sending data .Traditionally server are maintained by administrator ,with combine of operating systems hardware storage and different applications .Based on the service they serve we named for example serving email is named as email server , for serving file operation is named as file server .In traditional server if the storage is reached maximum storage capacity , the administrators responsible to add another server .Todays mostly as the technology is forward to cloud computing the server become more of virtualized . These kind of server hide server's software resource away from the hard ware on virtual server there are different operating systems storage and applications .Virtual server have different hosts [38]. The administrator is responsible to providing different service accordingly the user request we can make server virtualized either hard ware or software to create the virtual entities in cloud computing environment that entities in actually, are not physically present.

Virtualization work with one or many computing resource by combining or dividing resource to the end user that is similar operating environment there are three different fundamental abstraction which are memory, interpreter and communication link in computing system [39]. Operating system is responsible to manage computer resource for implementing these three abstraction that is virtualization make simple to their use by isolating users from one another and support replications and scalability for the systems. With four different mechanisms virtualization simulate physical object interface [1]

- Multiplexing. Create multiple virtual objects from one instance of a physical object.
- Aggregation. Different multiple physical objects create one physical object
- Emulation. Construct a virtual object from a different type of physical object
- Multiplexing and emulation

All cloud computing advantage is take over through virtualization for what serve to providers and consumers that are accessibly, reliability, security, hardware independency, isolation, hiding and disaster recovery and so on .after virtualization different users may have their own independent gust and host OS that can be run on the same HW done by virtualization layer [40] that is help to manage the complexity through well-defined interface. This interface is used to separate different levels of abstraction. Generally interactions with different sub system are minimize by layering that each sub system abstract through interface with other sub systems. In cloud this is performed by virtual machine monitor (VMM) or hypervisor, software that divide computing resource securely for one or more different virtual machine [1], lies between HW and OS.

There are different virtualization levels as stated in [38], this are

- instruction set architecture
- hardware
- operating system
- library support

Among this different layer there are interfaces to separate different levels of layer, the first interface is with the boundary of HW and SW is ISA (instruction set architecture). Most computers have two modes of operation: kernel mode and user mode [41]. The most fundamental software is in kernel mode (also called supervisor mode). In kernel mode you can access all hard ware and you can execute any instruction in machines like I/O which are forbidden to users the other mode which is user mode have only subset of machine instruction and user software are available . Instructions are divided in to privileged and none privileged which are execute in kernel and user mode consecutively.

The second interface is the application binary interface (ABI) that can access the hard ware with ensemble of application and library module. This interface is not able to access privileged instruction, in case, it tries use a system call.

The third interface is, the application program interface (API) [1]. It defines a set of instructions that the hardware was designed to give and execute the application access to ISA and also has HLL

library to invoke a system call. An abstraction for the code of an application at time of execution is known as process and a light weight process is known as thread. ABI is the projection of the computer system seen by the process, and the API is the projection of the system from the perspective of the HLL program. In Fig 9 which shows the interfaces among the software components and the hardware that an application uses library functions (A1), makes system calls (A2), and executes machine instructions (A3)

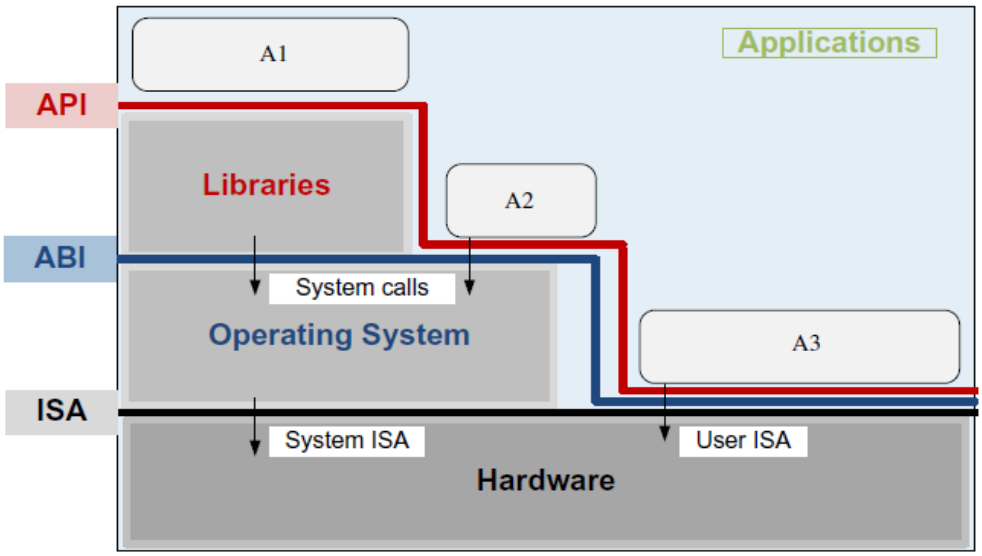

Figure 2-7:- Layering and interfaces between layers in a computer system. [1]

2.1.3.1. Types of virtualization level

2.1.3.1.1. Instruction Set Architecture Level

This done virtualizations at ISA level implemented by emulating in software [38]. in this ISA level approach , it is possible to run binary code written for various processors on any kind of HW machines .Instruction which is issued by gust VM is execute by emulator by translating native instructions and then executing them on the available hardware. Instructions like processor (add, sub, div), and the I/O (IN/OUT) instructions used for the devices. Emulation is done through code interpretation .Which is interprets the source instructions to target instructions one by one. When an emulator works by translating instructions from the guest to host, accomplishing the same task through instructions available on the host, without any stringent.

2.1.3.1.2. Hardware virtualization

Modern operating systems permit multiple processes to run simultaneously. Processors execute instruction with two different mode user and kernel, privileged instructions are run on kernel mode .Others are run on user mode. Hardware virtualization is done on top of bare hardware by creating a virtual hardware environment for a VM. Virtualization technique helps to map the virtual resources with physical resources.

2.1.3.1.3. CPU virtualization

A CPU is virtualized if it supports the ability to run the VM's privileged and unprivileged instructions in the CPU's user mode while the VMM runs in supervisor mode.

There are three techniques for handling sensitive and privileged instructions to virtualize CPU as stated in [42]

Full virtualization, VMM creates differentiate the environment between the guest and the host. In this techniques the operating systems directly can access all peripheral devices without knowing of virtualized environment and requirement modifications [42]

- . The VM is install a virtualization layer on top of the host OS which is responsible for managing all hardware. VM may run an applications, other application may also run with directly with host OS .some examples that use full virtualization are KVM and VMWHERE
- Para virtualization, is modifying the OS before it can be allowed to run in the virtualized environment as a VM. In which each VM runs on a slightly modified copy of the actual hardware. different reason are forced to use this para virtualization techniques [1]

    - Hardware may not be virtualized
    - For improving performance
    - For present simpler interface.

  Some example which uses para virtualization is XEN [42]
- Hardware supported virtualization, because of full and para virtualization are complicated we use this techniques the operating system also run in VMs without modification.

2.1.3.1.4. Operating System Level

OS level virtualization is that the virtualized layer, above the OS produces a partition per virtual machine demand which a copy of the operating environment on the physical machine .OS level virtualization creates different containers on a single physical server. The containers act as a real servers. This kind of virtualization is mostly used in creating virtual hosting environments to allocate hardware computing resources within different users. The VM share the hardware and operating system on the physical machine.

2.1.3.1.5. Library Support Level

Applications written mostly with HLL often call library modules. It also compiled into object code [1]. Applications are used APIs exported by user-level libraries rather than using lengthy system calls by the OS. User-level library implementations are designed to hide the operating system related details to keep it simpler for normal programmers since most systems provide an APIs, such an interface becomes another candidate for virtualization. By controlling communications link between applications and rest of systems by virtualization library interfaces through API. The software tool WINE has implemented to support Windows applications on top of UNIX hosts [43]

2.2.Related works

Cloud computing service providers deliver their resources based on virtualization with legal binding of SLA. In cloud computing, the amount of resources required can vary per user request. Providers also have to offer virtualized resources as per user request. To provide worldwide service, a provider should have data centers which are geographically distributed across world. When user needs to have some resource after grasping at any time the resource must scale up at any time when they need. Before that the providers must available enough resources as much as possible within near to user geographically and have manage the work load of data center. When we are seen resource scaling strategies we get two kind, vertical resource scaling and horizontal resource scaling. Most cloud providers are use horizontal scaling with its own advantage and dis advantage. In horizontal scaling, cloud provider take big advantage than cloud consumer which is paid more extravagance. The other scaling scheme is vertical resource scaling. This method also has its own advantage and disadvantage, that it is best for efficient resource utilization. When different resource allocation is working they are deal with resource scaling scheme, most resource

allocation is prefer with whether they are scaled or not. Most of provider use resource allocation methods with horizontal resource scaling methods. Even if some resource allocation scheme try to specify with auto scaling method, that is horizontal scaling with automatically scaled horizontally. In the following we try to see different literature views that are used horizontal and vertical scaling scheme.

In [44] is states that how to arrange large-scale jobs submitted to cloud aiming with job scheduling, and resource allocation in order to minimize cost which is a significant problem in cloud .They are try to work at application level for resource allocation. This scheme is dynamically allocate virtual resource to the cloud application. In order to improve resource utilization they are try to arrange their application with large scale job. The application may get resource by two means at running state and at arriving state this is choose by the cloud providers. To address multi-dimensional resource allocation problem by applying a resource allocation scheme using fewer nodes to process user's applications, they are use VM, when applications arrived the decomposed the application in to several types of jobs. For the exiting VM, they are assign jobs in running nodes to fair utilization. On working node, they are propose a framework for application allocation by virtual machines by combining resources to minimize work node. They try to dynamically allocate the virtual resources for the applications accordingly their work load by given in size smallest node. When a user starts an application, a set of VM that fulfill the minimum resource requirement for the application is given. This schemes is clearly show that they are used the horizontal scaling method that workload of the application increases virtual machine is newly assigned to specific application. From the above paper they have some draw back that when virtual machine is created for newly application or job it takes its own time and as they defined the algorithm they are doing by checking the capacity and add another node until they fulfil the size that application need, it also time taking job, it is not make decide automatically fixed size for specified job.

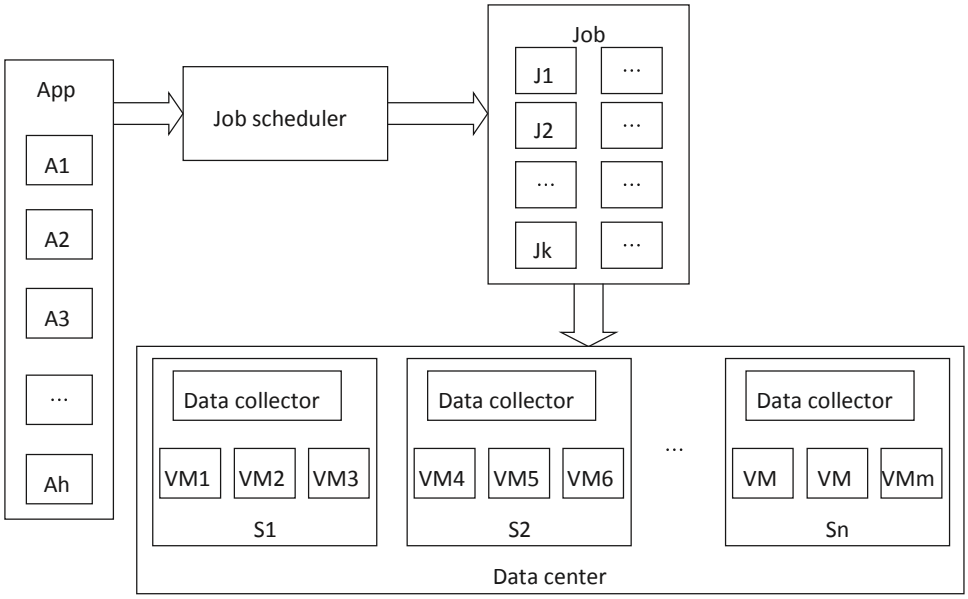

Figure 2-8:- Job allocation framework [44]

In [21] , they investigate vertical scaling for provision web applications and resolving CPU resource conflicts. They create elasticity controller uses a prediction model to perform proactive resource scaling. They combine changing CPU of virtual core with extending power of VM by plugging new CPUs on-the-fly that modern operating systems support CPU hot plug from an already running VM. In order to implement vertical scaling they combine capping and CPU-Hot plugin. They develop the architecture depict in Fig 2-9.

Initially VMs get some fixed number of CPUs with certain power unit. VMs have different priority. The goal is to have a better performance for high priority VM. If load increases, then virtual CPU speed of high priority VM increases too. Once CPU load reaches its limit, additional CPU is plugged to this VM. Other low priority VMs use extra cycles of physical CPUs. Clients with different priorities is beneficial for a provider, because existing capacity is utilized.VM running with high priority get high availability, by renting resources from low priority VMs. If high priority VM has additional resources, then they rent out these resources to low priority VM.

Their experiment consists of one computer which is used to run VMs and group of client emulator machines. Workload generator consists of 10 machines with 100Mbps network, which run client emulators. They used web traces of World Cup 98 to evaluate scaling controller with real workload

variation. having this from the above paper we identify the following drawback priorities virtual machine is assigned by provider ,but customer may have not any knowledge about the priorities virtual machine they may get extra cost since they pay for each CPU cycle and they may get over provisioning problem ,Application run on priorities virtual machine ,if suddenly it need more CPU cycle the un priorities virtual machines may not be run at the whole time ,There may be a number of un priorities virtual machine in case the priorities virtual machine need more VCPU They are not specified from which un priorities virtual machine should be taken ,When priorities virtual machine have more CPU cycle they can rent for other virtual machine in case the owner virtual machine application need unexpected CPU cycle can't take their own CPU cycle since they are rent , the have to be go other un priorities virtual machine ,Trash hold value is set by provider it is not automatic then resource scaling is occur when the predicted resource usage is above threshold but it must worked with automatically it must not be set by the provider.

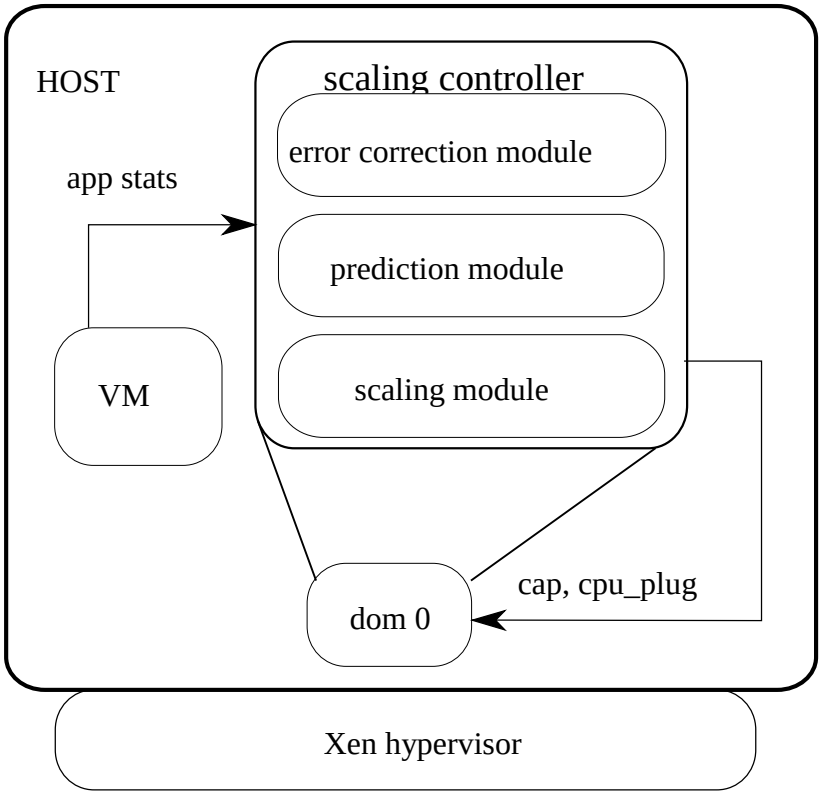

Figure 2-9:-Elasticity controller. [21]

In [45] states that, any cloud customers must have scale up and down their resource as they needs. They use virtualization technology to allocate resource dynamically within the applications demand. They are used skewness and virtualization techniques to support green computing they main aim of the study was that the capacity of a PM resource should be satisfied for all virtual machines resource needs running on it. , the physical machine number must be decreased as all need of virtual machine is satisfied ,so to minimize the energy the other idle must be turned off . They use virtualization to allocate resources based on the application demands and also they use skewness to measure the proportionality for server resource utilization. By combining different work load you can minuses skewness to minimizing skewness, they try to combine different types of workloads. To measure the skewness they are using based on hot spot and cold spot. Hot spot is used if any resource utilization is above hot threshold indicating the server is overloaded and then the VM migrate to away. The cold spot, resource utilization is below the threshold indicating the server is idle and also dynamically the power of server is turned off. This scheme is mainly work for horizontal scaling concerning for energy saving which is good for power consumption but they scale up and scale down with VM. they try to allocate resource whenever request is come they create one virtual machines and when work load is decrease they deallocate the resource from the virtual machine as we try to identify earlier creating VM it takes time for configuring the virtual machine so they are follow horizontal scaling method even if they try to make allocate dynamically, after allocating the resource they try to scale up and down horizontally. for making power saving sometimes the applications are moved from one virtual machine to the other at this time it makes performance problem in virtual machine migrating as it have its own advantage and disadvantages .

In study [46] stated that for pre-emptible tasks they create an algorithm that allocate resource adaptively. This scheduling algorithm for cloud system is used to adjust resource based on updating the actual task. It takes its own time to calculate the resource how much it may need .Mainly adaptive list scheduling (ALS) and adaptive min-min scheduling (AMMS) algorithms are used for scheduling the task. They also include static task scheduling, for static resource allocation, which is generated offline. They also work on repeatedly within fixed time online adaptive procedure for checking back and forth static resource allocation in each checking and evaluating process, scheduler calculated finished time of the submitted task then it assigned to the cloud. hear for every task it gives priority for the task that is already run in , but not given for new task because when it

adapt the execution environment it assign the resource more of to the already running task , this scheme also used horizontal scaling for each task they assign virtual machine .

In [47]they are try to study about scaling the resource vertically which is benefit to be in more of cost regarding the cloud resource type. Data centers can adapt to workload changes by scaling an application vertically. That vertical scaling has lower reconfiguration latency in comparison to horizontal scaling. They are looking different techniques like thresh hold, reinforcement learning, and parallel learning for predicting the workload and based on that they are looking for reserving resource. When they are looking threshold value and require expertise knowledge. When they are looking RL approach will adapt environment based on its own experience. With benefit of this method is ability to make decisions under uncertainty based on environmental observations with main drawback of dimensionality problem, in this study they create Vscaler using reinforcement learning algorithm with parallel learning algorithm. The agent in RL learns by visiting each of state which is a resource and action which is number of resource to be added or minimize in vertical scaling. In parallel learning the agent learn by working on individual task and periodically shares own observations with other agents. They are develop the architecture that how the Vscaler work, Then VScaler takes workload prediction and calculate the best resource allocation scheme for the next reconfiguration interval combining the RL and parallel learning scheme. Drawback of this work is, to predict the work load they try to combine with parallel learning and rain forcemeat algorithm , but they are not put any mechanism to when the prediction error at a time .

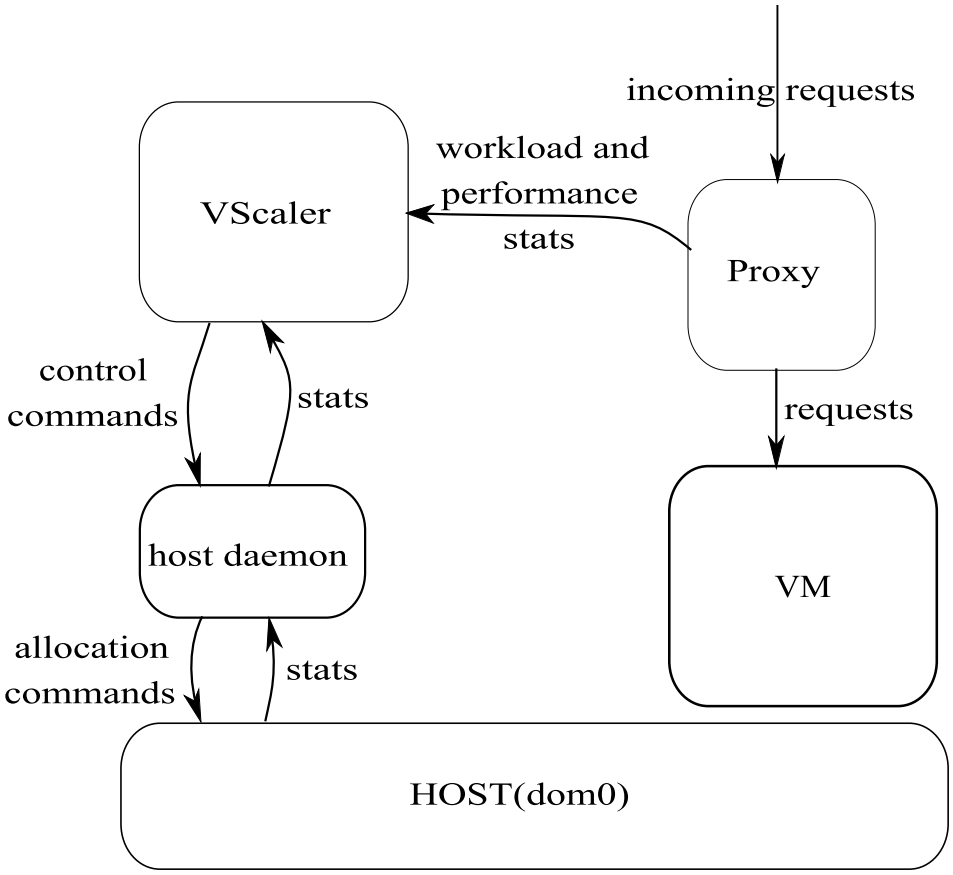

Figure 2-10:- Architecture of VScaler [47]

In [48]The authors propose SmartScale, taking into account both horizontal and vertical scaling in its resource management policy, i.e. by changing the number of virtual machines and changing the resources assigned to the allocated virtual machines. The authors claim that SmartScale reduces the total cost of running virtual machines and outperforms both traditional vertical and horizontal scaling policies. The combination of these policies brings a challenging problem that SmartScale must solve at runtime: determining the optimal scaling path and the trade-off of the corresponding reconfiguration. The authors solve this problem with a two-phase algorithm: computing the optimal virtual machine sizes and the corresponding application's throughput; and determine the minimum number of instances such that the total throughput is reached. SmartScale's scale algorithm also takes into account the information about the application provided by the customer.

Table 2-1: related work

Authore	Title	Drawback
Bo Yin, Ying Wang, LuomingMeng, XuesongQiu	A Multi-dimensional Resource Allocation Algorithm in Cloud Computing	➢ when virtual machine is created for newly application or job it takes its own time which is use horizontally scale methods ➢ they use statics when they need to create virtual machine ,it is not dynamic
Lenar Yazdanov and Christof Fetzer	Vertical scaling for prioritized VM provisioning	➢ priorities virtual machine is assigned by provider without knowhow of customer, ➢ priority virtual machine can fetch resource from non-priories virtual machines ➢ when more resource is required by non-priories virtual machines , there is no any means from where they take , ➢ if surplus resource are available in priories virtual machine , they have the right to rent these resource to other virtual machines ,in case after they rent to other , they may also fetch other resource from un priories virtual machine when they get need more resource ➢ Trash hold value is set by provider it is not automatic then resource scaling is occur when the predicted resource usage is above threshold but it must worked with automatically it must not be set by the provider
Zhen Xiao, Weijia Song, and Qi Chen	Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment	➢ This scheme is mainly work for horizontal scaling concerning for energy saving which is good for power consumption but they scale up and scale down with VM.

Jiayin Li, MeikangQiu, Jian-Wei Niu, Yu Chen, Zhong Ming,	Adaptive Resource Allocation for Pre-empt able Jobs in Cloud Systems	➢ hear for every task it gives priority for the task that is already run in , but not given for new task because when it adapt the execution environment it assign the resource more of to the already running task , ➢ This scheme also used horizontal scaling for each task they assign virtual machine.
Lenar Yazdanov and Christof Fetzer.	Vscaler :autonomic virtual machine scaling	➢ to predict the work load they try to combine with parallel learning and rain forcemeat algorithm , but they are not put any mechanism to when the prediction error at a time ,

From the above different researchers try to work on different mechanism to scale resource , most of them try to do their work with horizontal scaling which have low resource utilization , the others try to work with vertical scaling with priority virtual machines without knowledge of cloud users and giving the priority by adaptive the environment , whenever ones virtual machines have run at time , the new virtual machines may not be get the chance to scale the resource until they get more adapt the running environment .our thesis make able to scale vertically based on the threshold value and history data which is not in others works to get efficient resource utilization and efficient cost .

CHAPTER THREE: PROPOSED SOLUTION: VERTICAL SCALING

Form the above literature review we understand that most paper are works on resource allocating that's perform horizontal scaling. And some other papers have try to propose vertical scaling with different mechanisms like by assigning virtual machine priority and fetch this priories virtual machines resource from un priories one . The other one is try to scale vertically by predicting resource usage with reinforcement and parallel learning methods. These algorithm are takes their own time to configure the virtual machines whether vertically or horizontally. Hear we try to propose our algorithm which is more cost efficient compare to the other stated on related work

3.1. Algorithms

This algorithm mainly works with resource that is vertically scaled. We have done our algorithm by java server page and implemented this algorithm on OpenStack cloud management software. As depict in figure below, the architecture focuses on using the resource effectively.

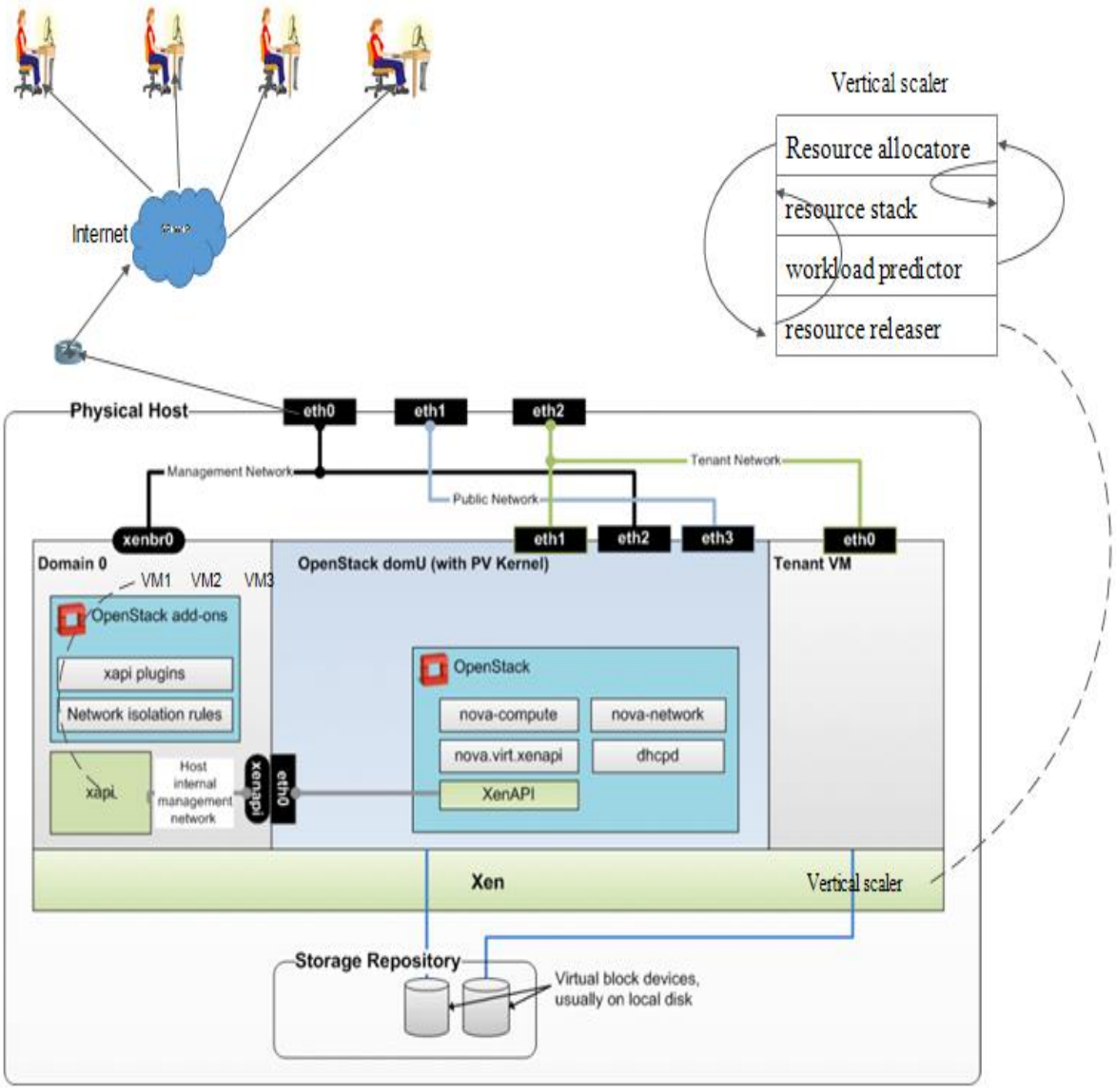

Figure 3-1 vertical scalar architecture

As shown in the above figure, we have a numbers of users to accesses some cloud data center to get resource. Whenever any kind of resource request comes to the data center, the adminstratore first differentiates which kind of resource is being requested; then it performs difeerent mechansims to allocate the requested resource the user requests on which the cloud data center offers. On the cloud, data centers have different servers. In our case, each server has to create any numbers of vertual machince with Xen hypervisour. We have used type one hypervisour on wich the hypervisour is installed with bare metal and our scalare is on the above hypervisor. Whenever the aplication need to run by users and users request more resource, this scaler is responsible to handle the request.

The scalers have 4 phases: work load predictor, resource allocatore, resource releaser, and resource stack. These all modules together work the resource scalling. We have been working on the application when there is more request; this time the scaler tries to up to the maximum size of the resource which is avaliable on the vertual machines. If it is above that specfic vertual machine, the scalre tries to fech other resource from other vertual machine which checks their stutuse that reserved cofficent variable for the next 2 hr in that machine. In our scalare, the work predictore works based on histroric data on that specfic vertual machine. When the application is instaledl on the vertual machine for first time, since the work predictore has no any kind of resource utlization information, it is forced to use CPU treshhold value to make scale up the machine. Having the number of applications vertual machines resource usage history for some 2 hour, the the scalere begins analaysing saved history resource usage data and reserve based on the reserve coffient able to give high resource utlization in that cloud providers . As we explain, earlier XEN hypervisiour is used credit scheduler. In the scheduler, each domain has two main properties: weight and cap. They determine the share of physical CPU time that the domain will get and the maximum of CPU time the domain can get respectively. It is widely configurable for the administrator.

Rresource allocater module is mainly based on the scheduler. It allocates the resource for the ppropriate usres. This allocater is mainly scale the resource based on the request up to the maximum size of resource that can be granted. For resource scalling firest, we have to allocate cloud resource, then we are going to scale accordingly. We scalled resource on the allocated resource is not enogh for handling the load. At this time, we are scalling the resource based on the application demmands. At nay time after the application use the apropriet resource, it must be released for others request uses. This is done by resource releaser module. To release the resource, it considers different thing. When users realeas resource automaticalliy the released resource added on resource stack . This resource stack mainly gives every information about currently avaliable resource in each vertual machines. The user can easly request based on the information in the resource stack. This basically helps in making decisionsn for users .

3.2. Resource allocation

In this, module we have allocated the avalible resource based on the information we have got from resource stack that every vertual machince resource usage statics. To allocate the resource, first the user must request the cloud resource to use over the cloud. Whenever the cloud provider gets this request, the cloud adminstratore must check the resource stack whether there is enogh resource or not. If yes, the cloud administrater can assign for the user based on the requested amount resource.

The cloud adminstrater is respnosible for forwarding any kind of request for the appropriate data center, he/she may see different option to select these data center beside the amounts of resource they have in each data center servers. When the user is the first to request in that specfic data center the adminstrater to configure vertual machine for all resource requestd. Whenenver the request is come, vertual machine resource is assigned to each users request until the maximum amount or numbers of resource to be scalled. As all cloud providers have specified their numbers of resource within range like tiny, small, very large, extra large and so on, users requst their resource based on these range, then vertual resource is allocated for it. When resource is allocated, and vertual machines work load is high , vertual machnes resource is scale in by feching requred resource from other neghbours vertual machines based on analysing reserve cofficient variable for the next 2hours. Every vertual machines resource usage statastics is registered on resource stack module to have net information about the usage of cloud resource . The hypervisoure has detail information about this resource usage statics on each vertual machines. The resource stack checks the availability of resources with virtual machine. If the resources are available on resource stack, the resource grants the access permission by the cloud administrator to use the resources to specific task. At any time, if the task require more resource based on the work prediction of the specfic task resource is scaled up and down. We are using hote plugin and cold plugin method for scale up and scale down resource.

3.2.1. Steps for allocating resources

- Cloud user requests for cloud resource to have virtual machines;
- Cloud administrator accepts this request by checking different pre-requests and sends for resource stack whether there is enough resource or not;

- Based on resource stacks information, the hypervisor begins assigning the requested resource;
- If the request is for the first time on the physical machine, then cloud administrator creates one virtual machine based on the request;
- If resources are available in resource stack and if resource request is done for new virtual machines and application, the cloud administrator is granted for the specific resource;
- If resources are available in resource stack and if resource request is done for the existing virtual machines and applications, the cloud administrator scale up the resource by fetching the resource from other virtual machine by analyzing reserve coefficient variable;
- If resources are not available in resource stack and if resource request is done for new virtual machines an applications, the cloud administrator diverts scheduler;
- If resources are not available in resource stack and if resource request is done for the existing virtual machines and application based on reserve coefficient variable, it will check again after 5 minutes until 30 minutes, since reserved coefficient variable is set for the next 30 minutes, if it is not, the cloud administrator scales up the resource by fetching the resource from other virtual machine by analyzing reserve coefficient variable from low priority task which compared with the request task;
- If resources are not available in resource stack and resource request is done for the existing task, the task has low priority the cloud administrator forced the task to the resource schedule and resource is scale down by giving the highest priority to continue the task whenever the resource is enough.

When we are allocating and scaling this resource, we are considering different factors to utilize the resource efficiently. When we try to allocate the resource, we have to analyze the usage and nature of resource that are getting from resource stack. Based on the information we gate from resource stack, we can allocate resource correctly to the hypervisor. This performance indicator for resource reduces execution time and save cost for customers. Resource Utilization Cost is one factor to make utilize the resource efficiently that depends on resource cost which is requested and the link communication is the cost for accessing the requested resource from the hypervisor to the user. Hopes are counted that response from the path, from which the resources will pass through. Our algorithm is working with the shortest distance to effectively serve the cloud users.

3.2.1.1. Algorithm 1:- User request

Globally initialize ←resource_to_be_allocate

Algorithm 1 User request

Input: - user_info

user

cloud_administrator

Output:- user_request(resource_to_be_allocate)

1. begin
2. users ← resource_to_be_allocate
3. loop:
4. cloud_administrator ← user_info return true
5. resource_allocate(resource_to_be_allocate)
6. goto loop
7. return user_request(resource_to_be_allocate)
8. end

3.2.1.2. Algorithm 2:- Resource allocation

Algorithm 1 Resource allocation

Impute :- user_request(resource_to_be_allocate)

resorce_stack(available _resource)

Output:- resource_allocate()

1. begin
2. i←0
3. user_request(resource_to_be_alocate)
4. if resource_stack(avaliable_resource) ←true
5. if user_request(resource_to_be_alocate) for new_vertual_machine
6. top1:
7. user_request(resource_to_be_alocate)←hypervisiour
8. loop:

```
9.          else if  resource_stack(avaliable_resource) >user_request (resource_to_be_alocate)
            && priorty_vertual_machine for user_request(resource) ← high
10.              user_request(resource_to_be_alocate +request_resourc)←hypervisiour
11.              priorty_vertual_machine for user_request(resource_to_be_alocate)←high
12.              resource_stack(existing_resource - resource_to_be_alocate)
13.              goto loop
14.      else
15.          if user_request(resource_to_be_alocate) for new_vertual_machine
16.            schedule(user_request(resource_to_be_alocate)) ← hypervisiour
17.           else
18.              loop2:
19.              if i < = 30 then goto top1, i←i+5
20.                 loop2
21.                  priorty_vertual_machine for user_request(resource_to_be_alocate)←high
22.                  if (i←30) then select less_priorty_vertual_machine
23.                        priorty_vertual_machine←high
24.                   resource_release(less_priorty_vertual_machine)
25.                    user_request(resource_to_be_alocate +request_resourc)←hypervisiour
26.                    resource_stack(avaliable_resource - resource_to_be_alocate)
27.
28.  end
```

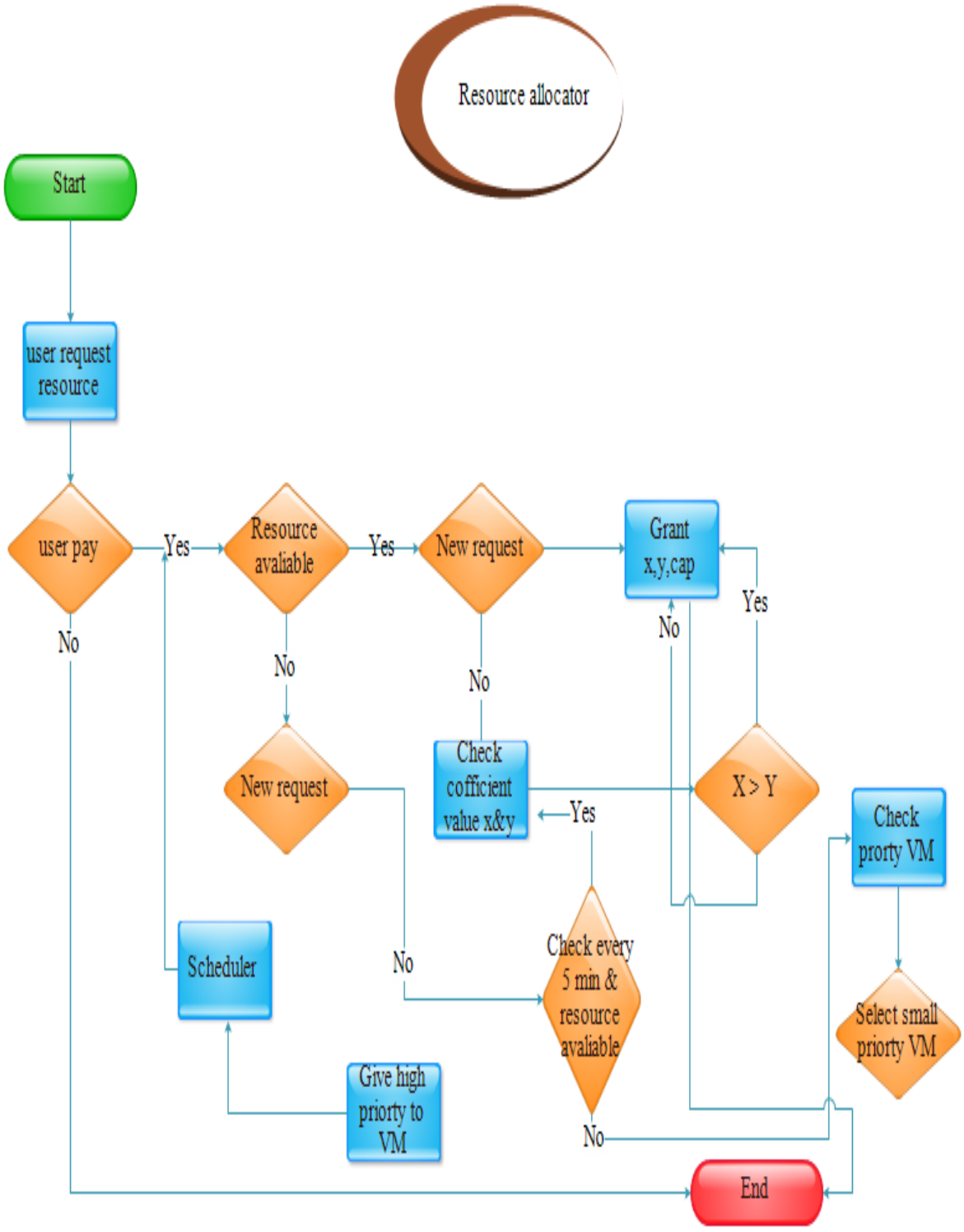

Figure 3-2:-resource allocation flow chart

3.3. Resorce stack

This algorithm is mainly used for cloud user to have all avalibale information about the cloud resource usage as well as the avalible resource at atime. As shown in table 1, when a user first requests the resource, cloud adminstarater sends this request to resource stack to check wether there is enogh resource or not for the time being. If there is enough resource, the cloud adminstrater works acordingly but if not he or she informs the user to use their option.

3.3.1. Steps for display cloud resource information in stack

- When request comes for the first time, the virtual machine is created with requested number of resource size.
- After creation of virtual machine, all numbers of resources are registered in resource stack based on applications request on virtual machines,
- Whenever the request comes from the users, the hypervisor look for surplus of resource from resource stack and the resource stack checks whether the request is from the new virtual machines application or existing virtual machine applications.
- If request comes from new virtual machine, and there is enough resource in resource stack, it begins to compare priority of virtual machine in resource stack with new virtual machine.
- If the priority of new virtual machine is greater than or equal to with other machine ,requested resource is granted but if not resource is not granted utile the priority become high
- Check for high priority virtual machine applications, whether there are enough resource available or not to continue the task. If it is, grantee the requested update resource stack
- If it is existing virtual machines applications request resource, it checks whether it is requested from the same server and users. It checks whether the applications is used before. If used and enough resource is available, resource is granted based on the requested and minimize the number of resource existing in resource stack; if not, request is canceled.
- If resource is requested from existed virtual machines applications but if there is not enough resource then by modifying the priority of the virtual machine to the highest priority, it waits until the grasped resource is left and granted. Here, the resource of

existing user virtual machines is released and sum up with the release resource sized with total existing resource sized in resource stack having remark that at any time when required sized resource is available; it takes to execute.

- If users finish their task based on SLA or any other interested to leave their resource at any time, cloud resource is released and update resource stack by sum up the release resource sized with total existing resource sized in resource stack.

As we have tried to define earlier, this resource stack is mainly used to store the used resource and to have the overall resource and application information in the cloud. By seeing this resource stack, cloud users can easily decide how many size of resource is rent and which resources have free at what time. And also, the cloud user mainly saves time and cost as the steps specifies no instance, or resource is forced as a whole without cloud user's knowhow.

3.3.2. Algorthm 3:- resource stack

Algorithm 3 resource_stack

Impute :- user_request(resource_to_be_allocate)
resorce_releaser()
resource_allocator(user_request(resource_to_be_allocate)
Output:- resource_stack(avalible_esource)

1. begin
2. resource_stack(avaliabe_resource)←virtual_machine1_available_resource + virtual_machine2_available_resource +..+ physical_machine_avaliable_resource
3. tope1:
4. resource_stack(available_resorce) ←resource_allocate(resource_to_be_allocate)
5. if resource_allocate(resource_to_be_alocate) for new_vertual_machine
6. if priorty_vertual_machine for user_request(resource_to_be_allocate) ← high
7. top2:
8. if resource_allocate(resource_to_be_alocate)<= resource_stack(available_resorce)
9. resource_stack()←resource(avliable_resource- resource_to_be_alocate)
10. else schedule(resource_allocate(resource_to_be_allocate)) ←hypervsiour
11. else goto tope1
12. select high priorty virtual_machien from resource_stack

13. goto tope2
14. else
15. resource_allocate(resource_to_be_alocate)>resource_stack(available_resorce)
16. resourse_release(resource_to_be_alocate)
17. user_request(resource_to_be_alocate +request_resourc)←hypervisiour
18. goto top2
19. end

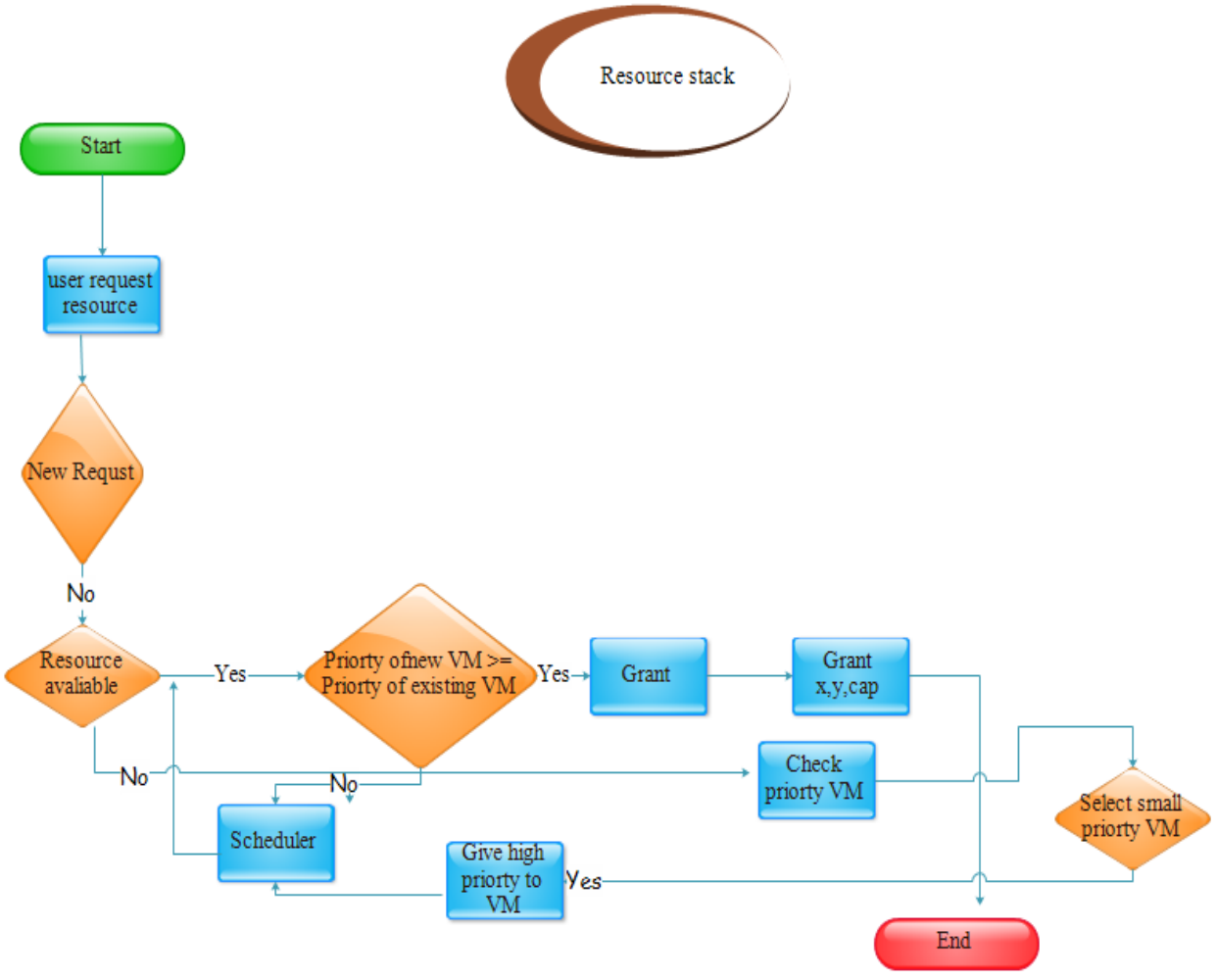

Figure 3-3: resource stack flow chart

3.4. Work load predictor

In [49] stated that the majority of websites will receive the most visits during the day, when the majority of people are awake. Many websites have busy periods during the day spikes in traffic just before school or work in the morning (7am to 9am), during lunchtime (12am to 2pm), and right after school or work (4pm to 6pm). Saying that, some websites will experience peaks of traffic throughout the day depending on the content they publish and the audience they target. For example, a children's website will be quiet at night, while a B2B website will be much busier during business hours. This algorithm mainly predicts based on historic data that may gather every state of virtual machine application resource usage. When the virtual machines applications is working for the first time, the prediction is done with CPU threshold value, threshold T_U=80. That is at which the CPU usage comes with 80 %, it adjusts the credit scheduler weight and cap values. This predictor predicts applications in every moment of activity. This is help to make somehow reserve resource based on the prediction. Whatever the virtual machines application first time threshes hold value is used for minimum of 2 hrs. And the value reaches at this threshold values, the cap and weight values must be adjusted, because until predictor gather these resource usage information for the sake of predicting the reserved resource for the next 2 hrs.

The algorithm is allowed to check the cloud environment in every moment; it registers the status of the CPU for doing prediction. If we have an application that has different users, we use that specific application to protect from other more uses. Our algorithm is doing this thing by analyzing statistics of the specified applications, then if in case we have more loads on that application, it reserves some amount of resource for each application. As we can see from the table blow, the status is taken every minutes for each user that uses that specified application, and then we get every cumulative summation of those users as well as for the applications. Every status is saved so that it is able to predict based on the historic data which may it looks at the back from the day to have analysis for specific time interval because some applications are over loaded at some time interval bounded.

Table 3-1: work load prediction pattern

Time	VM 1(1)	VM 2(2)	VM 3(3)
2:00-2:30	114.0	275.0	137.0
2:30-3:00	292.0	29.0	39.0
3:00-3:30	333.0	152.0	101.0
3:30-4:00	106.0	70.0	81.0
4:00-4:30	272.0	282.0	30.0
4:30-5:00	232.0	14.0	32.0
5:00-5:30	259.0	256.0	41.0
5:30-6:00	131.0	45.0	118.0
6:00-6:30	320.0	28.0	199.0
6:30-7:00	373.0	227.0	150.0
7:00-7:30	202.0	82.0	58.0
7:30-8:00	377.0	300.0	92.0
8:00-8:30	389.0	194.0	185.0
8:30-9:00	130.0	287.0	120.0
9:00-9:30	31.0	135.0	126.0
9:30-10:00	315.0	97.0	161.0
10:00-10:30	72.0	78.0	89.0
10:30-11:00	321.0	142.0	92.0
11:00-11:30	124.0	13.0	162.0
11:30-12:00	169.0	26.0	95.0
12:00-12:30	135.0	77.0	57.0
12:30-1:00	101.0	215.0	23.0
1:00-1:30	229.0	109.0	158.0
1:30-2:00	193.0	76.0	113.0

Let's t is time interval,

cp is cpu cycle ,

r is for rate between two consecutive cp ,

x is reserved resource from the 4 consecutive 30 mint rate ,

y is reserved resource from the 7conscutive day rate d

$r_n = cp_n - cp_{n-1}$ ----------equation 1

$x = (r_1+r_2+r_n)/n$ ----------equation 2

$y = ((d_n-d_{n-1})-(d_{n-1}-d_{n-2}) \ldots - (d_{n-5}-d_{n-6}))/n$ ----------equation 3

3.4.1. Steps for predicting work load for applications

- First, we use threshold value for predicting CPU executions if we have not any kind of historic data in resource stack , T_U=80, and adjust cap and weight value
- When resource is assigned for virtual machines applications, work load predictor begin to check CPU usage which is register on resource stack at every moment
- When every 10 minutes analysis, the load for the CPU usage for specified virtual machine application, it sums up the result and puts the average.
- And also for every two hour, it put the sum average of CPU usage for the user's applications.
- It comparing current CPU usage with previously 12 consecutive 10 minute ,
- If it shows increment, then the predictor lists the rate for the next two hrs.
- It checks the status of 7 consecutive days; if it has these historic data, before for the two hour with different time interval and compares this 7days average CPU load for the user's applications.
- If the average of 12 consecutive 10 minutes status is greater than average of each consecutive 7 days, then the resource allocator module is called for reserving CPU cycle based on equation 2 before resource is exceed.
- If the rate is 0 or –ve number, it will compare the 7 consecutive day's rate and if the average of the 7 consecutive days is less than or greater than the predictor call resource allocator module with reserving resource based on equation 3.

3.4.2. Algorthm 4:- Workload peredction

Algorthm 4 Work load predictor

Imput :- user_request(resource_to_be_allocate)

resource_allocator(user_request(resource_to_be_allocate)

Output :-workload_predictore(reseverd_resource)

1. begine
2. th← threshold value for CPU usage , cap←caps value
3. n ← maximum values in some event←0, i←time_interval←0
4. x←reserved coefficient rate for a momonet ,y←reserved cofficent rate for aday
5. rst[i]←cpu usage for every i minute, cp←current cpu values ,index_in← aray index

```
6.      index_min←index for sum up minute ,index_min_hr←index for sum up hour
7.      sumhr←sum of eery hour , sumrst←sum of eery minute
8.    if user_request(resource_to_be_alocate) for new_vertual_machine
9.      top1:
10.     if th>=80 then resource_allocate(resource_to_be_alocate + cap)
11.   else
12.     loop:
13.       for no_virtual_machine reaches to n value
14.       for  no_user reaches to n vlues
15.       for i<=1400  then rst[i] ←cp , i←i++
16.       goto loop ,every_ten_minute_interval←10
17.     loop1:
18.       if index_in ←1400
19.           loop2 :
20.             for index_in reaches every_ten_minute_interval
21.             sumrst+←rst[index_in], goto loop2
22.     minute_sum[index_min]=sumrst , sumrst←0,
23.      every_ten_minute_interval+←10,
24.      index_min++←index_min
25.      goto loop1
26.     index_min+←12
27.       loop3:
28.         if index_in_hr ←140
29.           loop4 :
30.             for index_in_hr reaches every_twelve_minute_interval
31.              sumhr+←sumhr[index_in_hr], goto loop4
32.           minute_sum[index_min_hr]=sumhr
33.     sumhr←0,
34.     every_twelve_minute_interval+←12,
35.     index_min_hr++←index_min_hr
36.     goto loop3
37.       if x >0 && y>0
```

38. if x>y then resource_allocate(resource_to_be_alocate+x)
39. else resource_allocate(resource_to_be_alocate+y
40. if else x >0 && y<0 then resource_allocate(resource_to_be_alocate+x)
41. if elase x<0&&y>0 then resource_allocate(resource_to_be_alocate+y)
42. top1:
43. else resource_allocate(resource_to_be_alocate)
44. goto top1
45. end

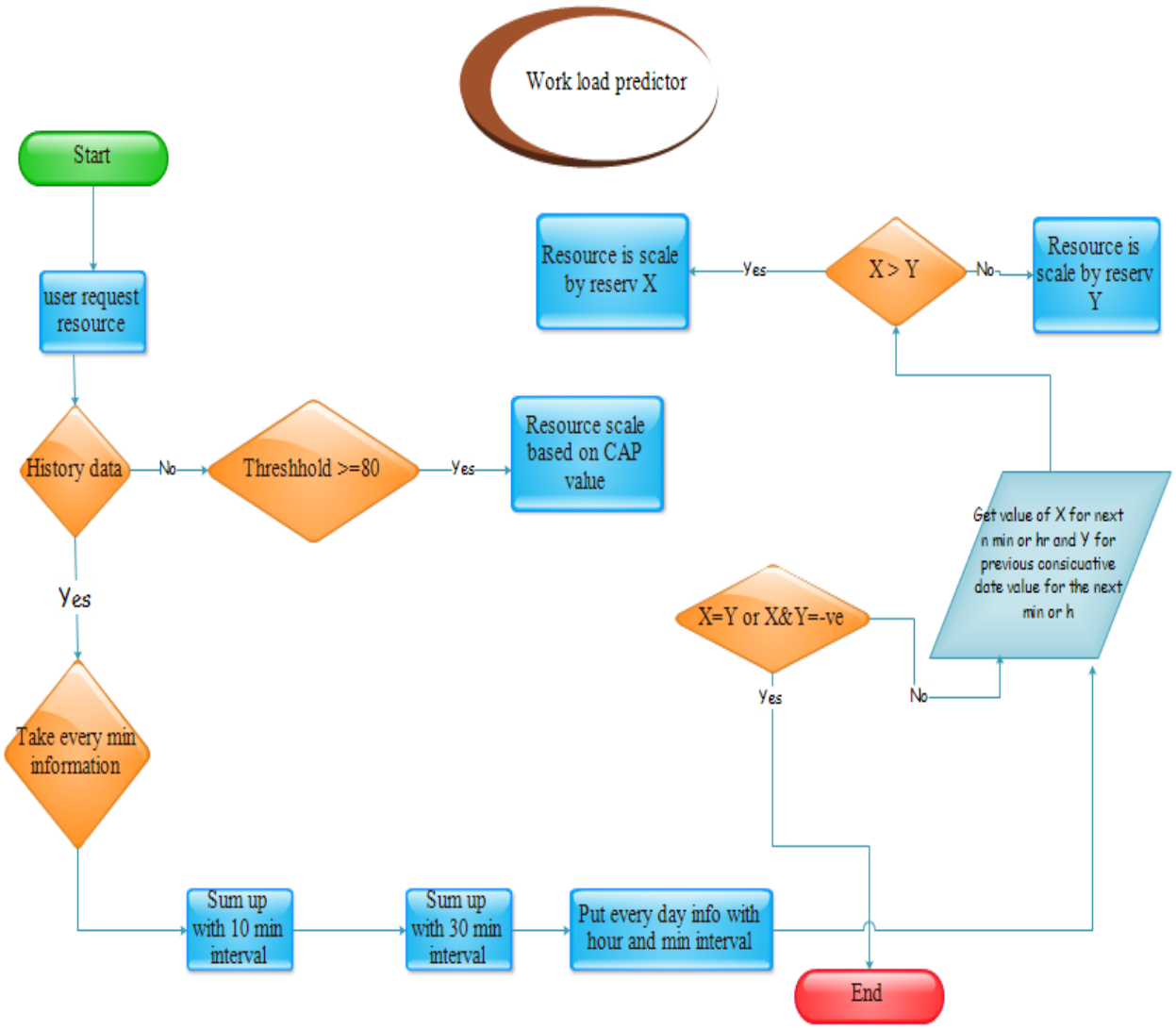

Figure 3-4: work load prediction flow chart

3.5. Resource release module

When users finish the job, the resource should release to resource stack module. Before releasing the resource, it checks the payment status of the resource. This module works if users need to release the resource and if the rent time is over. Resource is released when users finished their work and when users need more resource; and on stack there is not enough resource, it releases the grasped resource to protect resource starvation.

3.5.1. Steps for resource releaser

- Resource allocator and resource stack send request for resource release;
- If request come from the users to leave the cloud, it makes post check whether the necessary payment is done or not;
- If the resource is used based on the payment, they are paid then the resource is released and add update the resource stack;
- If resource release request comes from the resource allocator, release the resource and update the resource stack;
- If resource release request comes from the resource stack, release the resource and update the resource stack

3.5.2. Algorithm 5 resource releaser module

Algorithm 5 resource releaser

Input:
user_request(resource_to_release)
resource_allocator(U(R)),
Output: resource_release()

```
1. begin
2.      user_request(resource_to_release)
3.        if user_request←true
4.             resource_stack(avalible_resource+resource_to_release)
5.         elsle
6.             masage_to _user
7. end
```

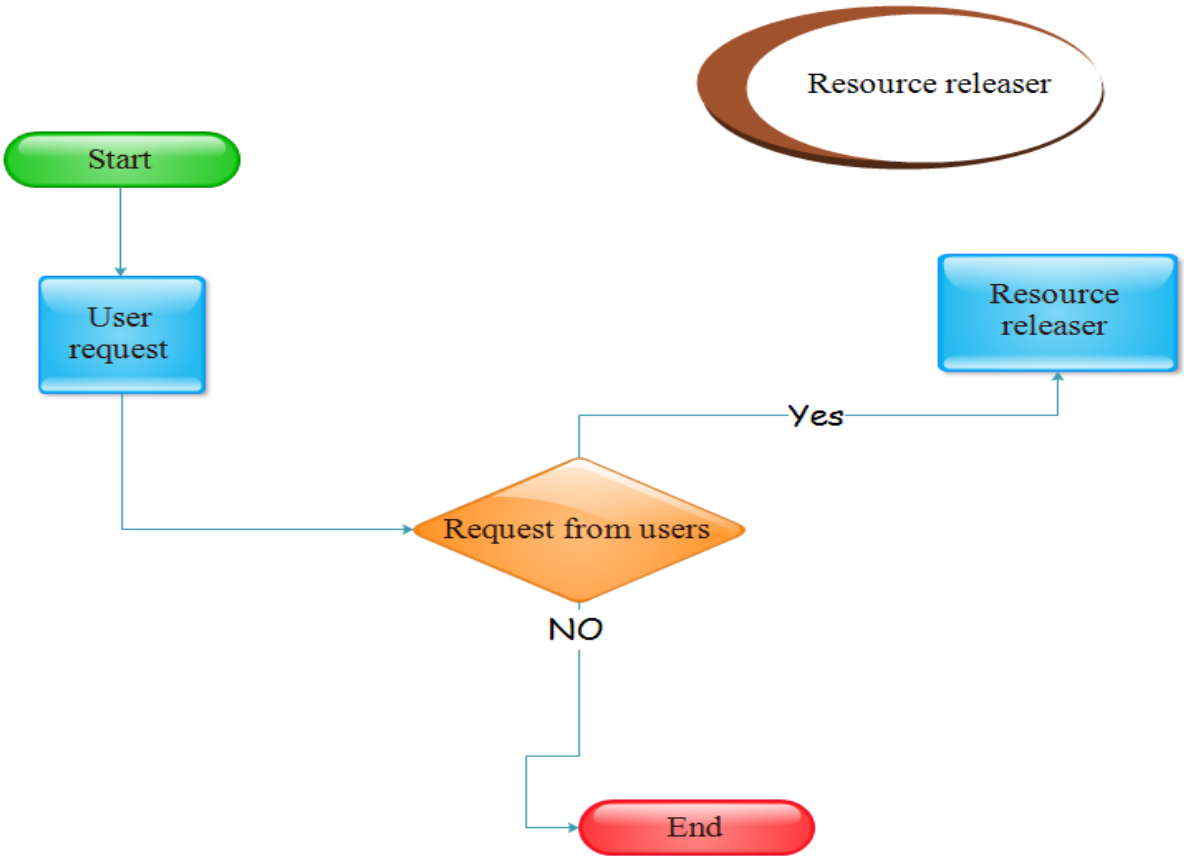

Figure 3-5: resource release flow chart

CHAPTER FOUR: EXPERIMENTAL SETUP

4.1.Experiment setup

Cloud computing is a typical for allowing on-demand access to a shared pool of resources. These resources are mainly shared on virtual machine. As cloud resources are very precious providers, consumer must be very careful in using them. Resource shouldn't be over provision or under provision. Providers should serve appropriate resource according to users request and also user must use it accordingly even if they have no clue about the usage of resource. Cloud environments enable the hosted Internet application to quickly cope with the spikes in workload. Resources scaled either horizontally or vertically, when overload is coming the provider make dynamic scaling the resource; this dynamic in most providers is making able to the horizontal scaling but dynamically it makes. Here, the most well-known providers do not use vertical scaling. Most of them prefer to be a horizontal resource scaling. We prefer to do our thesis with cloud software management tool OpenStack with branch kilo. We are eager to able this cloud software management with future vertical scaling with a new approach from earlier vertical and horizontal scaling.

As we define in chapter two, we try to analyses different scaling methods as we try to explain the horizontal scaling has its own disadvantage as high increasing virtual machine and also host. It makes very complex; it takes huge space and also it takes more time to configure from 2 min to 5 min. Our work mainly focuses on in enabling a cloud vertical scaling scheme which is not supported by most cloud providers. This scheme mainly makes advantage for both provider and customer. Most importantly, it is good for customers as far as cost is concerned. The customer are making with les extravagancy because it is not time taking in creation of virtual machine. Our work is more of customer side that focuses on how could the customer's decreases cost when using cloud resource. This horizontal scaling is a waste of one of the precious thing in cloud: resource. The provider is responsible to manage this resource effectively, as most customers may have not known how to use resource; they may mislead to use the precious resource. That is why the providers make sure all information must available to customer, and then the customers may choose as what they need. We are doing our experiment within 4 computer naming computing, controller, network, and storage node. Within each node, we install OpenStack cloud management software.

4.2.Research software and hardware tools

The following are the tools used for the accomplishment of this research:

4.2.1. Software requirement

- Operating System : Linux
- Technology : Java and J2EE
- IDE : My Eclipse
- Web Server : Tomcat
- Database : My SQL
- Cloud platform : OpenStack
- Virtual machine :XEN

4.2.2. Hardware requirement

An establishment of cloud environment has been done with OpenStack cloud platform with 4 personal computers namely controller, network, and compute and storage node. These four personal computers have their own resource listed as follow:

- Controller node :- 3.3 Ghz, 4 GB RAM ,500 GB disk, 1 NIC with 100 Mb/s for management interface
- Network node :- 3.3 Ghz, 2 GB RAM,500 GB disk, 3 NIC with 100 Mb/s for management ,tunnel network interface and for external networking
- Compute node:- 3.3 Ghz, 4 GB RAM,500 GB disk, 2 NIC with 100 Mb/s For management and tunnel network interface
- Storage node :- 3.3 Ghz, 4 GB RAM,500 GB disk, 2 NIC with 100 Mb/s for management and tunnel network interface

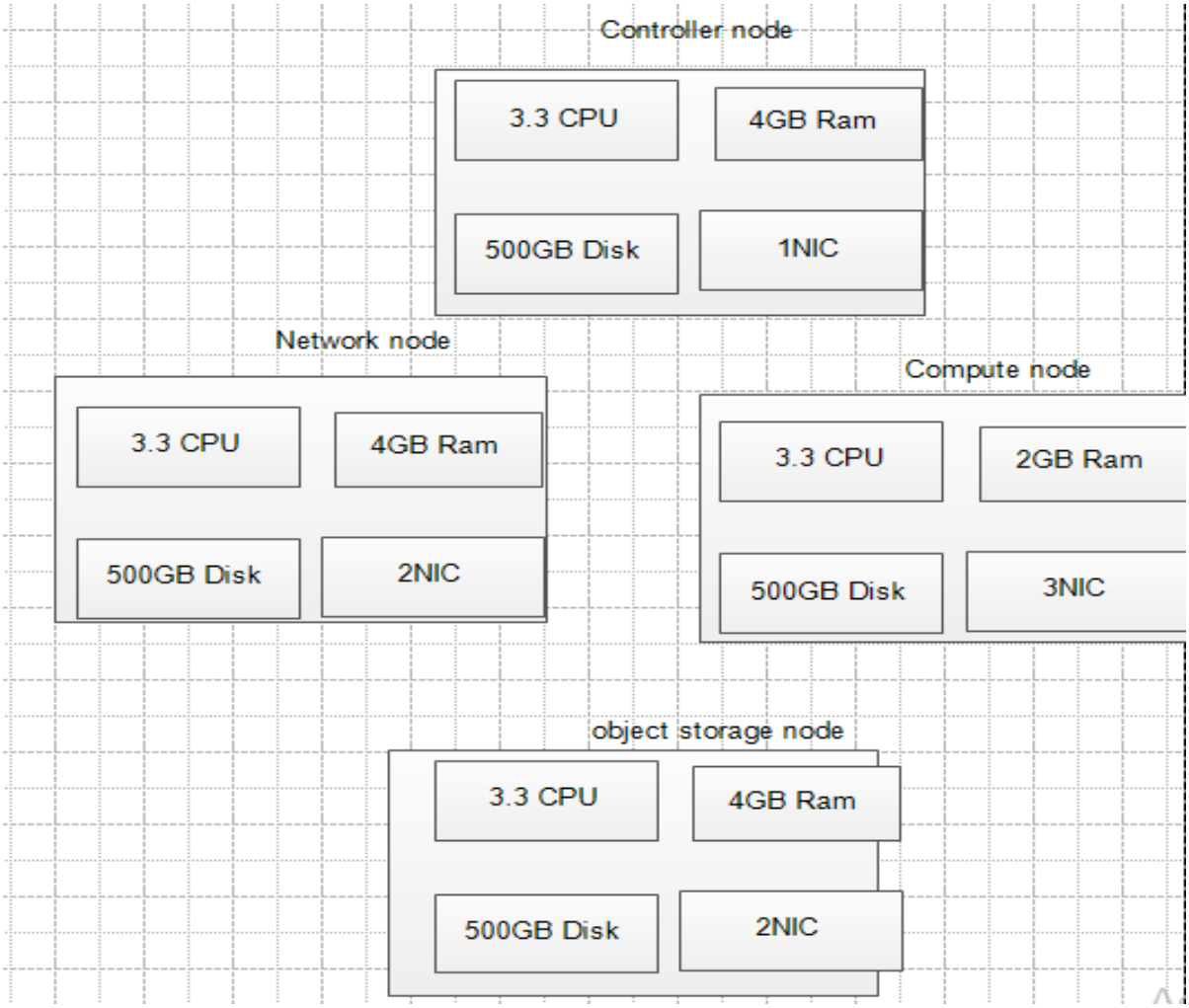

Figure 4-1: lab setup

4.2.3. Virtual machine

Today we have different virtual machines in different platform [6], XEN, KVM and VMWHERE. Generally, the considered hypervisors support vertical scaling with all considered resources. Today operating system support hot and cold plunging which you can add and remove resourcing under running without any effect. Virtual machines are support different operating system (listed in Table 4-1).

Table 4-1:-Comparative of different hypervisors

	KVM	VMware	Xen
CPU core Add/ Remove	Yes No	Yes No	Yes Yes
Memory Add/ Remove	Yes Yes	Yes No	Yes Yes
Disk Add Remove Extend Shrink	Partly ? No No	Partly Yes Yes Partly	Yes Yes Partly No
Network add Remove	Yes No	Yes Yes	Yes Yes
Para virtualization	No	No	Yes
Full virtualization	Yes	Yes	Yes
Host CPU	X86,x86-64,IA64,PPC	X86,x86-64	X86,x86-64,IA64
Gust CPU	X86,x86-64,IA64,PPC	X86,x86-64	X86,x86-64,IA64
Host OS	Linux	Proprietary Unix	Linux, Unix
Gust OS	Linux, window, Unix	Linux ,window, Unix	Linux, window, Unix
Live migration	Yes	Yes	Yes

From the above table, we analyses that XEN hypervisor is good for our thesis because XEN support hot and cold plugging, which is not in KVM and VMWHERE hypervisor. And also XEN hypervisor is support both full and para virtualization.

4.2.4. Future of OpenStack

This OpenStack is a model with lot of related service that other open source software doesn't have. This makes OpenStack more secure because of having many related service; the security is stronger than the other open source cloud software. The other issue with OpenStack is high availability with storage that has no with other open source cloud management software every object are placed with three disc for reliability purpose , if ones disk goes down with disaster or power fails, data will not lose because of replica the object with different disk. Because of the above reasons, OpenStack platform was selected to deploy cloud environment. This OpenStack cloud software is used on Linux operating system and it uses virtual machine KVM and XEN as a hypervisor. OpenStack is developed with python programming language with some java script.

4.3.Control node

We set up the controller node ruining with different services from OpenStack such as the identity service (keystone), image service (glance), compute management, network management, block storage management, telemetry management, telemetry agent, networking plug-in, the dashboard (Horizone), SQL database, message queue, and Network Time Protocol (NTP), orchestration, object storage proxy service. We also configure the controller node with hardware 3.3 Ghz, 4 GB Ram, 500 GB disk, 1 NIC with 100 Mb/s for management interface. This management interface requires a gateway to provide internet access to all nodes for installation package and updating different services. Since we have different communicating program ruining on different node, thus we try to install NTP to synchronize services among different nodes to reference more accurate server and the other nodes to reference the controller node. We also used a message queue that provides asynchronous communications protocol which the sender and receiver of the message do not need to interact with the message queue at the same time. Messages placed onto the queue are stored until the recipient retrieves them. We use RabbitMQ message queue service to coordinate operations and status information among services on controller node.

- Keystone service

  On the controller node, we are setting with identity service for user permissions and giving available service within their end point. This identity service have different component like user which are any person, system or service that used OpenStack. This service validates any incoming request made by users. After login the user are assigned token to accesses

resource this users assigned directly to token (An alpha-numeric string of text for accessing OpenStack APIs and resources which is revoked for finite time) to access resource and also can be directly assigned to a particular tenant (it is a stack used to group or isolate resources.it also may map to a customer, account, organization, or project). We also identify service database having different attribute and then identify service is installed on controller node. The keystone service still listens on ports 5000 and 35357. We configure the apache server within the listed port for client and administrator one. Having this adjusting the ownership and give permission for every request by configure authentication token and end point URL. Then we create each service entity and API endpoint; by default OpenStack use 3 variants for each service, Admin, internal and public. The admin API endpoint allows modifying users and tenants by default, while the public and internal APIs do not, OpenStack supports multiple regions for scalability then we configure with one region. The Identity service provides authentication services for each OpenStack service. The authentication service uses a combination of domains, tenants, users, and roles. For every project, we set an administrative project, user, and role for administrative operations in our environment and same to for demo and public one. Using open RC client environment script, the identity service interacts with OpenStack client. These scripts typically contain common and unique choices for clients. Then whenever we need to use the environment, we need to load the script with either admin or demo or whatever you create to use it. Generally, we use the following steps to install key tone in our controller node:

- Installing and configuring NTP
- Installing kilo branch OpenStack
- Installing and configuring database server
- Installing and configuring rabbit massage queue service
- Creating a database named keystone and granting all privilege for the specified database name
- Installing keystone on the node and configuring the database to make connection , the memcach service , configure the UUID token provider and Memcached driver
- Configuring apache HTTP server

- Designing the authentication token
- Configuring the end point URL
- Creating service entity and API endpoint
- Creating project, users and roles
- Creating OpenStack client environment script
- Loading the client script.

This client script is loaded at any time in which role you need to run the project or the service then you can use the defined service.

- Image service

  The OpenStack service that is installed in controller node next to identity service (keystone) is image service (glance). The OpenStack Image service (glance) enables users to get, register, and retrieve virtual machine images through REST API that enables us to query virtual machine image metadata and retrieve an actual image. We store our virtual machine image through this image service. OpenStack image service accepts different API requests for disk or server images. And from end user the image metadata also accepted and also OpenStack Compute components is accepted as well. By default, the OpenStack image service have different component; among them glance-API is for accepting image API for retrieval and storage purpose. Glance registry if for Stores, processes, and retrieves metadata about the images and lastly database to store images data. As we do in keystone service, we create database named glance and service credentials for accessing the glance with specified users, and API endpoint which is for public admin and internal for specifies the region also. As we create in key stone service as we create one region their hear we also create one .then we are install image service in this controller node and make appropriate configuration to use the database reserved for this service and we try to give the authentication for this glance user both on glance-API and on glance-registry, we upload Ubuntu 14.04 ISO image with QCOW2 disk format, *bare* container format, and public visibility on glance service to access by any project. After this installing and configuring image service in controller node we continue to install and configure compute service

- Compute service

  This OpenStack service is mainly used to host and manage cloud computing systems. This service interacts with the above listed two OpenStack service keystone for authentication and image service, for disk and server image and additionally with horizon for users and administrative interface. This service have different component; nova-API service is for sending and receiving to different end user compute API calls. This service has different activities such as for running instance. We use this API, nova-API-metadata service accepts metadata requests from instances.

  Mostly we use this service when we are try to run in multiple host modes with nova-network installation, since we are using this scheme. Nova-compute service is a worker process to create and terminates virtual machine instances through hypervisor API,since we are using KVM-API; it accepts actions from the queue and performs a different system commands. We try to launch XEN instance and updating it in database through this service. Nova-scheduler service is the other component that takes VM instance request from queue and determines on which to compute server be host it; nova-conductor module is a mediator between nova-compute service and database for protecting direct accessing of cloud database. Nova-network worker daemon is for accepting and manipulating networking tasks. It performs different network tasks. Nova-consoleauth daemon is for authorizes tokens, nova-spicehtml5proxy daemon, nova-xvpvncproxy daemon which are used for providing proxy accessing running instances through a VNC connection for Supporting browser-based novnc clients., for provides a proxy for accessing running instances through a SPICE connection. Supports browser-based HTML5 client and for provides a proxy for accessing running instances through a VNC connection.in compute for image management, we use two component that are nova-object store daemon, euca2ools client.

  The nova-object store is an S3 interface for registering images; it is used primarily for installations that must support euca2ools which talks to nova-object store in S3 language and then nova-object store translates S3 requests into Image service requests. The nova client is used to enables users to submit commands as a tenant administrator or end user. Having the above detail information, we have tried to install in this node the compute service in that first we create the database named with nova, give the admin credentials to gain access and create the service credentials by creating the service user , giving the admin role for those service user, and by nova service entity at the last we create the endpoint API

for computing service for accessing via public, admin ,and internal for specific region, Next, we mount compute in controller node by installing nova-API, nova-cert, nova-conductor, nova-consoleauth, nova-novncproxy, nova-scheduler, python-novaclient. We configure the database and massaging quos for accessing purpose. We also configure the identity key, VNC proxy to use the management interface IP address of the controller node, the location of the Image service. And finally we are populating the computer data base and restarting all nova service to finish our compute configuration in this control node. Like nova-cert, nova-consoleauth, nova-scheduler, nova-conductor, nova-novncproxy . After we setup all necessary setup on compute node, we made a list of different service component to assure that successful launch and registry of each service.

- Networking service

  This networking allows you to create and attach interface devices managed by other OpenStack services to networks. Different Plug-ins are used for accommodate different networking equipment and software, that are used for deployment purpose. This neutron service have different component to completion the required work. Neutron-server is one of component for Accepts and routes API requests for the appropriate OpenStack Networking plug-in for action plugs and unplugs ports, creates networks or subnets, and provides IP addressing. For networking installations to route information between the neutron-server, agents and DB to store networking state for particular plug-ins, we use massaging ques. Neutron manages all networking problems for the physical (PNI) and virtual Networking Infrastructure (VNI).

  To create firewall, load balancer, VPNs and the neutron enable tenants. It also provides the networks, subnets, and routers. As we try to identify this neutron service gives all networking futures as a physical setup, it has also one external network to represents a view into a slice of external network accessible outside the OpenStack installation. External network IP addresses are accessible by anybody physically on the outside network. This neutron service also has one or more internal networks, with this internal network instance are launched. Only the VMs internal network, or subnets coupled through interfaces to a similar router, can access VMs connected to that network directly. To have these all accessibility, we install and configure neutron service in controller node. First as we do

before to accesses the above configured service, here also we create database, service credentials, and API end point. We create a database named neutron and with the admin credentials; we create neutron users and give admin roles to named neutron users; we also create service entity named neutron, and finally we create networking service API endpoint.

All the above producers are a pre request to install networking component in this controller node. After we finish to install these component, we install networking component named neutron server which acts as this control node as a server and the other node to be as client. We configure the neutron to connect the database with the service, the massage ques, the identity service, to enable the modular layer 2 plugin overlapping IP and routing service. This plug-in uses different agent like open*vSwitch* to build the virtual networking framework for instances. We also configure the ML2 plugin file on neutron service to enable the *flat*, *VLAN*, generic routing encapsulation, and virtual VXLAN type drivers, GRE tenant networks, and the OVS mechanism driver. Here, we also reconfigure compute to manage the network with neutron service by giving authentication through keystone on controller node; finally we populated the database and restarting the computer and networking service.

We create virtual network infrastructure to which the instance connects the external network and tenant network. The external network typically provides Internet access for the instances we created. We tried to enable internet access for individual instance by public IP address using for accessing multiple tenant, first, we create this external network. Then on this external network, we tried to create subnet with public IP giving with range and the getaway as well. The tenant network provides internal network access for instances. This tenant network also needs its own subnet attached to it; we give with DHCP for instance to obtain IP address. We create our routers and internal network having launch different instances for our purpose. We launch one virtual machine to access this VM resource as when the demand increase we automatically scale up usage of resource with means vertically scale up.

- Dashboard (Horizon) service

  It is a Web interface that enables cloud administrators and users to manage various cloud resources and services. This dashboard enables web-based interactions with the Compute cloud controller through the APIs. Having all installation and configuration all the above necessary OpenStack service, we install and configure this dashboard (Horizon) service on the controller node.

- Block storage (Cinder) service

  It provides persistent storage to the virtual machines and provide an infrastructure for managing volumes, and interacts with Compute to provide volumes for instances. This service have different component. Like other service, it has an API called cinder this API Accepts different API requests, and routes them to the cinder-volume for action. This cinder-volume, interacts directly with the Block Storage service, and processes through a message queue such as the cinder-scheduler. The cinder-volume service responds to read and write requests sent to the Block Storage service to maintain state. Cinder-scheduler diamond selects the optimal storage provider node on which to create the volume. The cinder-backup service provides backing up volumes of any type to a backup storage provider. To have the above listed component in controller node, first like the other service; we create the database named cinder, create the service credentials to create cinder users and giving admin roll and create cinder service entities. We create the Block Storage service API endpoints. After doing this procedure, we install and configure the cinder service in control node. We configure this service by connecting the cinder database created, earlier, give authentication to these cinder users to have full access, and then by doing populated the block storage database

- Object storage (swift)

  The OpenStack Object Storage is a multi-tenant object storage system. It is highly scalable and can manage large amounts of unstructured data through a RESTful HTTP API. It has different component like swift-proxy server that accepts service API and raw HTTP requests to upload files, modify metadata, and create containers. It also serves file or container listings to web browsers. To improve performance, this component uses optional cache that is usually deployed with mecache. Swift-account-server component is to manage

accounts defined with service; swift container-server manages the mapping of containers or folders within the service. This service is also installed in control node that handles requests for the account, container, and object services operating on the storage nodes. Like another service in OpenStack, we are not going to create a database in thin node because each storage node we are going to create SQLite database, then we directly going to have create a swift user, giving the role for this created users, and finally creating service entity and API endpoint. Now it is the time to install and configure the swift service in this controller node. Before starting the Object Storage services, we create account, container, and object rings. The rings determine where data should reside in the cluster. There is a separate ring for account databases, container databases, and individual objects but each ring works in the same way. These rings are externally managed, in that the server processes themselves do not modify the rings. Each ring is created and configured in the controller node. In account ring, the account server processes and handles requests regarding metadata for the individual accounts or the list of the containers within each account. This information is stored by the account server process in SQLite databases on disk. In container ring, the container server processes and handles requests regarding container metadata or the list of objects within each container. Like accounts, the container information in stored as SQLite databases. In object ring, the object server process is responsible for the actual storage of objects on the drives of its node. Objects are stored as binary files on the drive using a path that is made up in part of its associated partition and the operation's timestamp. The timestamp is important as it allows the object server to store multiple versions of an object. The data and metadata are stored together and copied as a single unit. Then we finalized our installations by restarting the swift service; we installed in this controller node.

- Heat service

  This service is mainly used for creating and managing cloud resource. This service has different component like heat command-line. Client is a CLI that communicates with the heat-API to run AWS cloud formation APIs. It processes API requests by sending them to the heat-engine over RPC and finally heat-engine orchestrates the launching of templates and provides events back to the API consumer. This service is installed and configured in controller node as we have done previously in other OpenStack service. We create first

database named heat, and then we create heat users; giving role for those created user, create heat stack owner role and give this role for heat users for managing the stack. We create service entity and service API end point, and then we install and configure the heat service in controller node. Finally, we populated the database and restart the heat service to make sure we are installing and configuring well.

- Ceilometer service

  This service is used to metering data related to OpenStack services. It collects event and meters data by monitoring notifications sent from services. It has different component like ceilometer-agent-compute are runs on each compute node and survey for resource utilization statistics , ceilometer-agent-central to survey for resource utilization statistics for resources not knotted to instances . To scale service horizontally, multiple agents can be started, ceilometer-agent-notification goes on a central management server and chomps messages from the message queue(s) to build event and metering data; ceilometer-collector run on central server and dispatches collected telemetry data to a data store or external consumer without modification. Ceilometer- alarm-evaluator runs on one or more central management servers to define when alarms fire based on a threshold over a sliding time window. Ceilometer-alarm-notifies runs on one or more central servers to permit alarms based on threshold evaluation for a collection of samples and ceilometer- API to provide data access from the data store. Having the above component, we tried to create and configure a mongo DBdatabase, service credential and API endpoint of telemetry service for this controller node. We also create the ceilometer database, create a ceilometer users, and give the role for this created users. WE also create the service entity and endpoint API for the service ceilometer. Finally, we install and configure ceilometer package in controller node. To have an update system, we restart all installs ceilometer service in this controller node. After this, we are going to install different service in different nodes.

4.4.Network node

Networking in OpenStack gives a tenant-facing API for defining network connectivity and addressing in the cloud. This network is a virtual network service that provides a powerful API to define the network connectivity and addressing used by devices from other services like nova. Here, in OpenStack network, we can create more topology by creating and configuring networks

and subnets to attach virtual devices to ports on these networks. OpenStack networking uses plug-in, implementation of the OpenStack Networking API.

As a controller node, we are going to setup different OpenStack service in network node like OVC agent, networking ML2 plugin, L3 agent, DHCP agent, NTP and the OpenStack package itself. We also configure the network node with hardware 3.3 Ghz, 2 GB Ram, 500 GB disk, 3 NIC with 100 Mb/s for management, tunnel network interface and for external networking. Like controller node, we create the management interface that requires a gateway to provide Internet access to all nodes for installation package and updating different services. In network node, we also try to install NTP to synchronize services. After installing all the same service with controller node, we are going to install and configure the neutron package, authentication mechanism, message queue, and plug-in in the network node. Then, we install and configure the ML2 plug-in uses the OVS mechanism agent to build the virtual networking framework for instances. It is a framework allowing OpenStack networking to simultaneously utilize layer 2 networking technologies found in multipart data center. DHCP agent is configure in this node for VMs automatically receive IP addresses through, and we configure metadata agent for provides configuration information of credentials to instances.

4.5.Compute Node

We do the same procedure here in compute node to install and configure common service in controller node. The compute service relies on a hypervisor to run VM instances like KVM,XEN and so on. Here, we use XEN hypervisor in this compute node as explained in chapter one. This OpenStack compute, in this node, is to host and manage cloud computing systems. It interacts with OpenStack identity for authentication. We are going to setup here different OpenStack service in network node like hypervisor, openvswich agent ,networking ML2 plugin, compute service, NTP, telemetry agent and the OpenStack package itself .We also configure the compute node with hardware 3.3 Ghz, 4 gb Ram ,500 gb disk, 2 NIC with 100 Mb/s for management and tunnel network interface or management. The management interface requires a gateway to provide internet access to all nodes for installation package and updating different services. After installing all the same service with controller node, we are going to install and configure compute package, and then we restart the service to have full functionality. We also configure compute node to use networking since ML2 plug-in the *OVS*mechanism (agent) to build the virtual networking

framework for instances. We also install ceilometer-agent in this compute node for notifications and an agent to collect compute metrics.

Generally, we configure these five different nodes with OpenStack cloud software management for our experiment to implement our algorithm. With this experiment, we are eager to show how the cost of customer subsides by vertical resource scaling, by making fixed tubules instance and by paying what the time the customer use. With these three attributes, cloud customers can save his money from extravagancy, and also the cloud providers also make high resource utilization. Our algorithm takes all the aforementioned issues into consideration.

CHAPTER FIVE: IMPLEMENTATION AND RESULT ANALYSIS

In this chapter, we describe and analyze of our findings concerning scaling resource which is vertically we give a cost analysis based on our implemented solution in comparison with other infrastructure solutions. Additionally, we compare our proposed solution with some other solutions.

5.1. Implementation

We are set our experiment with OpenStack cloud platform, we are trying to make OpenStack scale resource vertically whenever the application demand is high. This make more minimize cost of resource usage for cloud user and highly utilize the resource for cloud provider. For our experiment we are used **web** application .we are using JMTREE for load intensive increase and decrees to the system. To emphasize that the scaling behavior of a cloud system is based on scaling of underlying resources,

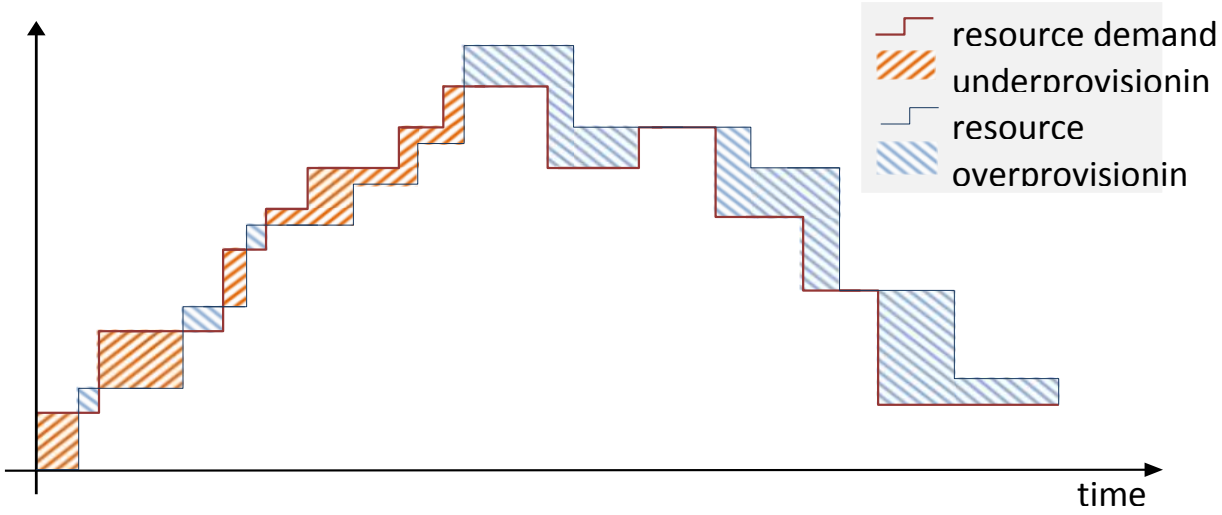

Figure 5-1: resource provisioning

From the above graph we understand that over provisioning and under provisioning is not efficient resource utilization, resource is automatically scaled up based on the resource available vertically. Resource utilization is used for both cloud provider and customer as we discussed on chapter one, more of for cloud customer to minimize cost usage.

For our experiments we have used a XEN virtual machine hypervisor. Since it is an open source virtualization monitor as we try to explain in earlier its advantages than other. It is referred as a para-virtualization hypervisor to manage the un-privileged domain (guest) from the most privileged domain (dom0). Xen has many types of scheduler to manage the virtual machine running on CPU. The major schedulers are [50]

- Barrow Virtual Time (BVT),
- Simple Earliest Deadline First, (SEDF),
- ARINC653 scheduler and
- Credit scheduler

.

The XEN hypervisor support for allocating CPU implemented using the default credit scheduler in XEN hypervisor. To distribute CPU processing power among virtual machines, Xen considers each virtual machine similar to a process in operating systems: it switches CPU among its virtual machines after each time slice. This process is called VM scheduling. Xen's default credit scheduler allows each managed virtual machine to have a guaranteed a portion of CPU's processing power with different parameters: weight, cap, vcpu, and time slices.

- Weight indicates the priority for CPU computing power of a virtual machine, relative with the others. It determines the share of physical CPU time that the domain will get

- *Cap* is used to define the upper threshold of the CPU load for a virtual machine, given in percentage.

- *vCPU* is the number of virtual CPUs allocated to a virtual machine.

- *Timeslice* specifies the duration of each virtual machine's execution before the scheduler interrupts it and switches to another virtual machine. This value can be tweaked to change the responsiveness of virtual machines.

For our experiment, we defined Scale up threshold $T_U=80$. In our experiment we increase a load for VM1 when CPU usage is 80 % it analysis and take from VM2 resources, then it scale up to full fil SLA.

We create 3 different virtual machine and we install different application which is running on virtual machine , for the first time we are set CPU values as a threshold whenever it reach at this point it scale up as we try to explain on the algorithm , after gathering information it go to predict based on the history stored

In our implementation the resource stack is created using CPU pool utility in XL tool stack. initially virtual machines resource(vCPU,vRam,vIO,vBW) are e shared all the physical resource(pCPU,pRam,pIO,pBW).base on this data cloud user can get all avalibale information about the cloud resource usage as well as the avalible resource at atime. Initially request comes for specific virtual machine, availably of resource is checked its threshold value. In this stack every virtual machines resource are grouped and put in to one pool. The physical machine resources are unplug and grouped in to pools based on the virtual machines request.

In XEN there are credit schedules which have cap and weight value, initially it set by 0 libvirt package is used to interact with the hypervisor. It has the ability to manage virtual machines on XEN. The request made by OpenStack is converted into XEN managed configuration through libvirt API interface.

Self-analysis of particular VM can be done provide by XENTOP to get VM resource utilization statistics. It will get the information based on some time slice value produced.. All the components are implemented and run on top of the XEN hypervisor. By default the scheduler gives equal priority for each threads (vCPUs) running on pCPU. On resource statics if it has information it began generate reserve resource value and generate predicted values then it began reserve resource When there is reserved value , it is more sharing by vCPUs, automatically scheduler reduces the CPU cycles of each virtual machine running on the pCPUs. This make by hot plugging, without any effect of other virtual machines states but when it did this it consider also other virtual machines reserved resource values .The main factors for scaling CPU virtually are, initial number of virtual CPU should be defined for launching virtual machine, The maximum no of virtual CPU required for scaling at running state. The total number of virtual resource can’t be exceeded the total number of physical machine resource

As we try to create 3 virtual machines VM1, VM2 and VM3 of vCPU1, vCPU2 and vCPU3 with 400, 300 and 200 cycle, shared the pCPU. Here scheduler gives equal priority for all the running virtual machine. Every CPU utilizations statics are register,

5.2. Experimental results

For the first time the scaling up action is invoked when the specified metric remains above the specified threshold value for a specified number of time periods gathering useful information history data ,if it is , it reserve resource by analyzing virtual machines resource usage statics to will have more load prediction virtual machines . This is to ensure that a scaling up action is not triggered due to a sudden spike in the value of a metric. It should have patterns. To optimize scaling it is quite important to study and understand the traffic load on the application in a production environment. As on [49] states that the majority of websites, will receive the most visits during the day, when the majority of people are awake. Many websites have busy periods during the day spikes in traffic just before school or work in the morning (7am to 9am), during lunchtime (12am to 2pm), and right after school or work (4pm to 6pm).

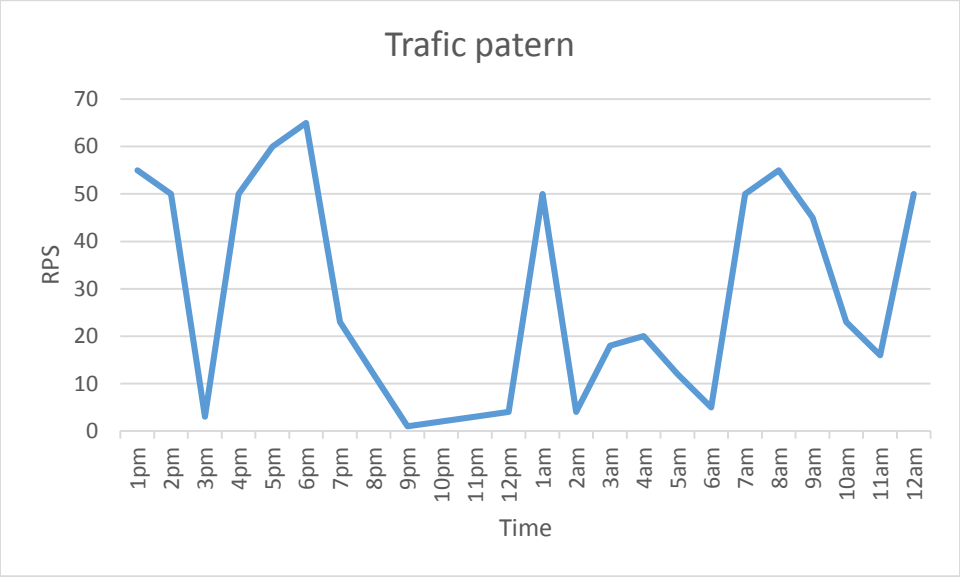

Figure 5-2: traffic pattern with time

Creation of virtual machines in cloud, our main objective and in our experiment we focused on the scaling the resource. Any virtual machine is launched it should have the configuration parameter max vCPUs value is equivalent to maximum physical processor value. Once the value is fixed we cannot upgrade the CPU level. Our algorithm is executed on top of the XEN hypervisor once the virtual machine is created on the availability of virtual resource, the load increased to the VM using JMTREE which consumes more CPU to show our experiment on VM1.

To determine scaling up it is very important to determine the appropriate threshold value for the selected metric if the virtual machine is for the first time, then we leave this threshold value if we have historic data, it will scale up based on the calculated reserved resource before the virtual machine is down. A low threshold will result in underutilization of the virtual machines resources. On the other hand, a high threshold may result in higher latency, hence degrading the end-user's experience. Therefore, we define the threshold value by considering the throughput corresponding just meeting the SLA for the application.

Load of the three virtual machine are

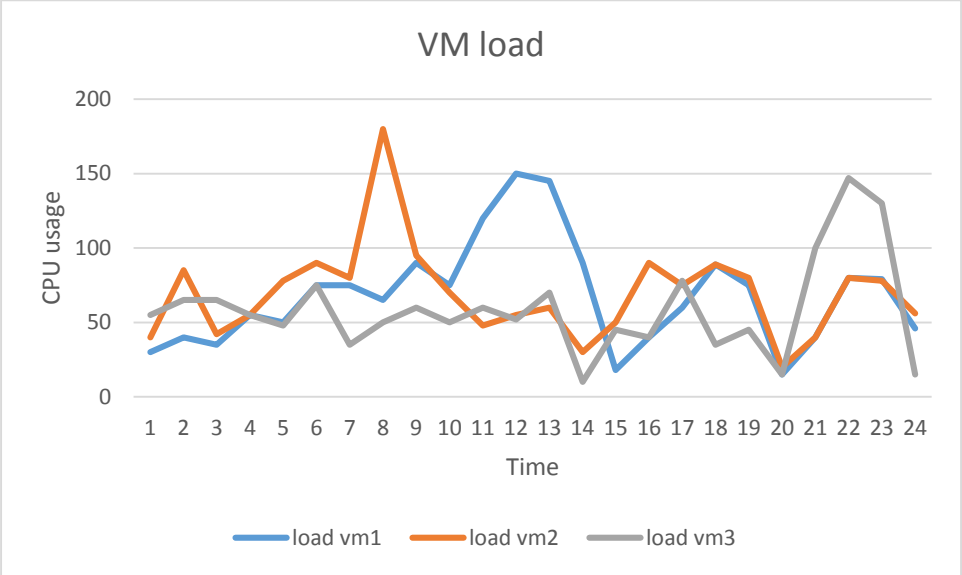

Figure 5-3: work load for VM

As shown from the above diagram load of different virtual machines that are run on XEN server to have run on application , hear is by setting CAPs value manually the administrator done which is adjusting for the first time it give priority for VM2 since its threshold value is greater than 80% , resource is comes from VM1 and try to continues VM2s works as well as when vm1 work load

is high priority is given to the this virtual machine , the resource is fetched from vm2 and vm3 and give to this specific virtual machines

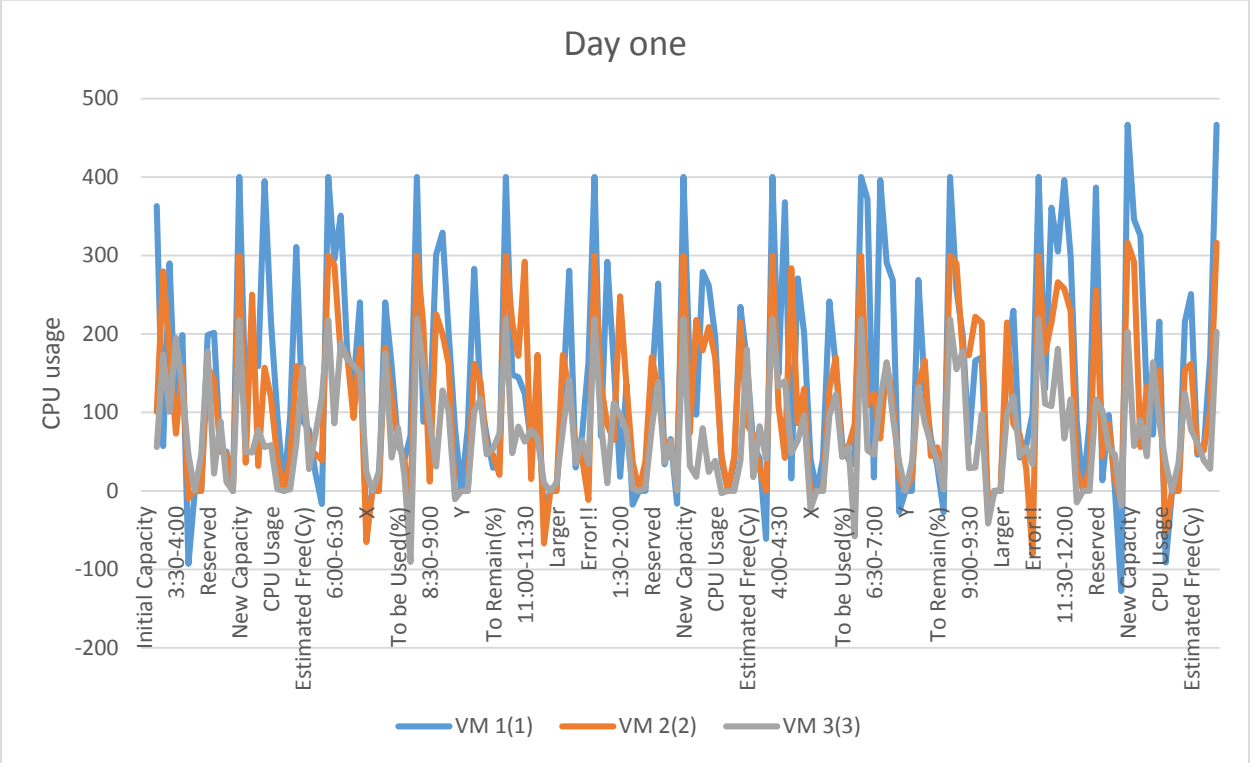

Figure 5-4: day one resource utilization

The above graph time 2:00-4:00 the 2 virtual machine VM1 and Vm2 doesn't need to add more resource but in VM3 it comes with 88.9 which is above thresh holed , when VM3 try to reserve resource for the next two hour, the reserved coefficient is become 46. Which is the current execution rate for this virtual machine 65% of 200 cycle , when we reserve to more 46 cycle to this ., the prediction is become 88.9 % will execute , since it is above thresh hold it reserve more resource add 17 CPU cycle for the next two hour, at this time the other 2 virtual machine have not used their resource which is assigned to them it get this surplus resource from these two virtual machines since their current execution is 49% for VM1 and 52 % for VM2 , and also , the predictor tells that there is no more load . . at a time 6-8 VM3 has predict for the next two hours , that need more extra resource ,result analysis it resource usage be 80.5 % of the previously 217 cycle , become 219 cycle , two more cycle is reserved as VM run its application without any interruption ,at night time 10-12 time VM1 predict 96.7 % of 400 is used for next two hours ,based on the coefficient value , since it is above threshold value it need more of 16.7% of then the new resource

be 466 cycle. From day one CPU usage we analyses the total prediction error is 0.521%, 0.142 % and 0.171 % for VM1, VM2, and VM3 consecutively.

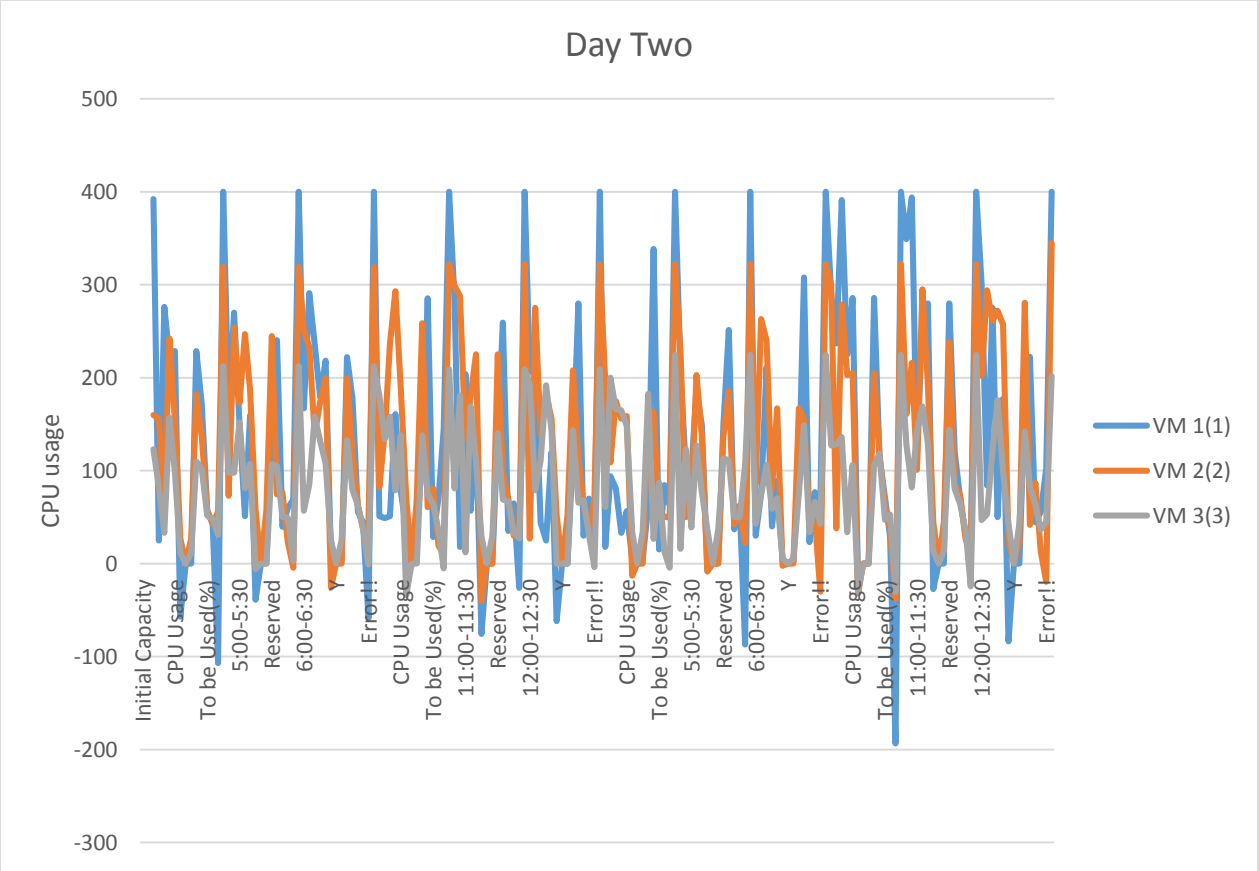

Figure 5-5: day two resource utilization

From the above based on previous date final two hours resource information second day is begin to predict, hear it use two reserved coefficient variable for hour and day rate values, as the graph illustrate at time 6-8, VM1 has 438 cycle and VM2 has 299 cycle, which is predicted based on previous 2 hour from 4-6 with reserved coefficient variable 2 and 19 consequently. VM3 has predicted for next two hour 83.2% of 162%, since on VM3 CPU reduce from 200 cycle to 167 cycle. Since VM3 has above threshold value then the resource exceed to 167% which is 3.2% of 162 cycle. from time 10-12 VM2 threshold value is above 80% ,2.6% is increased , previously it has 299 cycle, reserved coefficient for the two hour was 50 cycle , it become 82.6% to be used for the next two hour , then VM2 capacity is scale in to 307 for time 12-2 . Other virtual machine be used as it is without scale in, and on time 12-2 has passed new capacity for the day 3 execution.

Day 3 work with the information based on day 2 resource scale in .From day two CPU usage we analyses the total prediction error is 0.131 %, 0.034% and 0.104% for VM1, VM2, and VM3 consecutively.

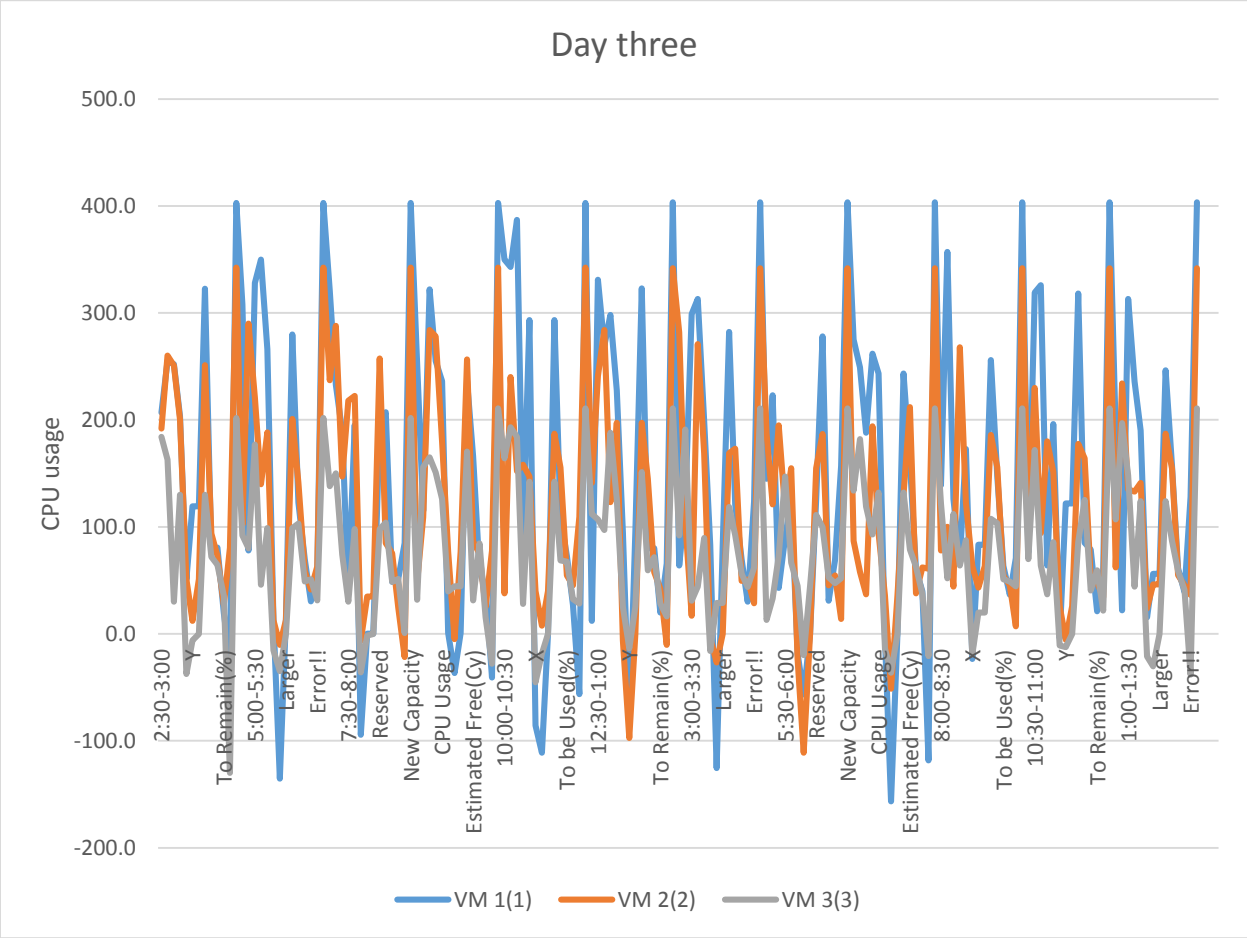

Figure 5-6: day three resource utilization

Day three analysis show that it take information from day two, from time 4-6 VM1 has get threshold 91.2% CPU usage be for the next two hour 6-8 , this done by reserving coefficient rate for two hour , and also it check coefficient rate for previous date form 6-8 , when it checks the date rate coefficient value. VM1 was scale in before 403 cycle, here for next two hour 11.2% is exceed from threshold based on reserved coefficient values .it scale in with 11.2% of 403 cycle which is scale to 448 cycle, VM2 also predicted to be 79 % of 300 cycle so it has not any effect on it, VM3 has 109.7 % which is predict for next two hours from 6-8 execute, so it scale in 29.7% of 212 cycle which is scale to 274.9 cycle is reserved for next two hour. From the time 10-12 VM2 indicate

threshold value to 86.6%, which is 6.6 % of 251 cycle which is the capacity for time 10-12, is increased for next two hours from 12-2 which is 267 cycle .for next two hours also the predictor analysis that VM2 has need more resource based on reserved coefficient variable 17.3% of 267 cycle, since its above 80% threshold value, 97.3% it become 313 cycle is reserved for next two hours. this done as usual without stop or shut down the running VM, total prediction error in this day is 0.369 % ,0.305% ,0.148% for VM1, VM2 and VM3 respectively .

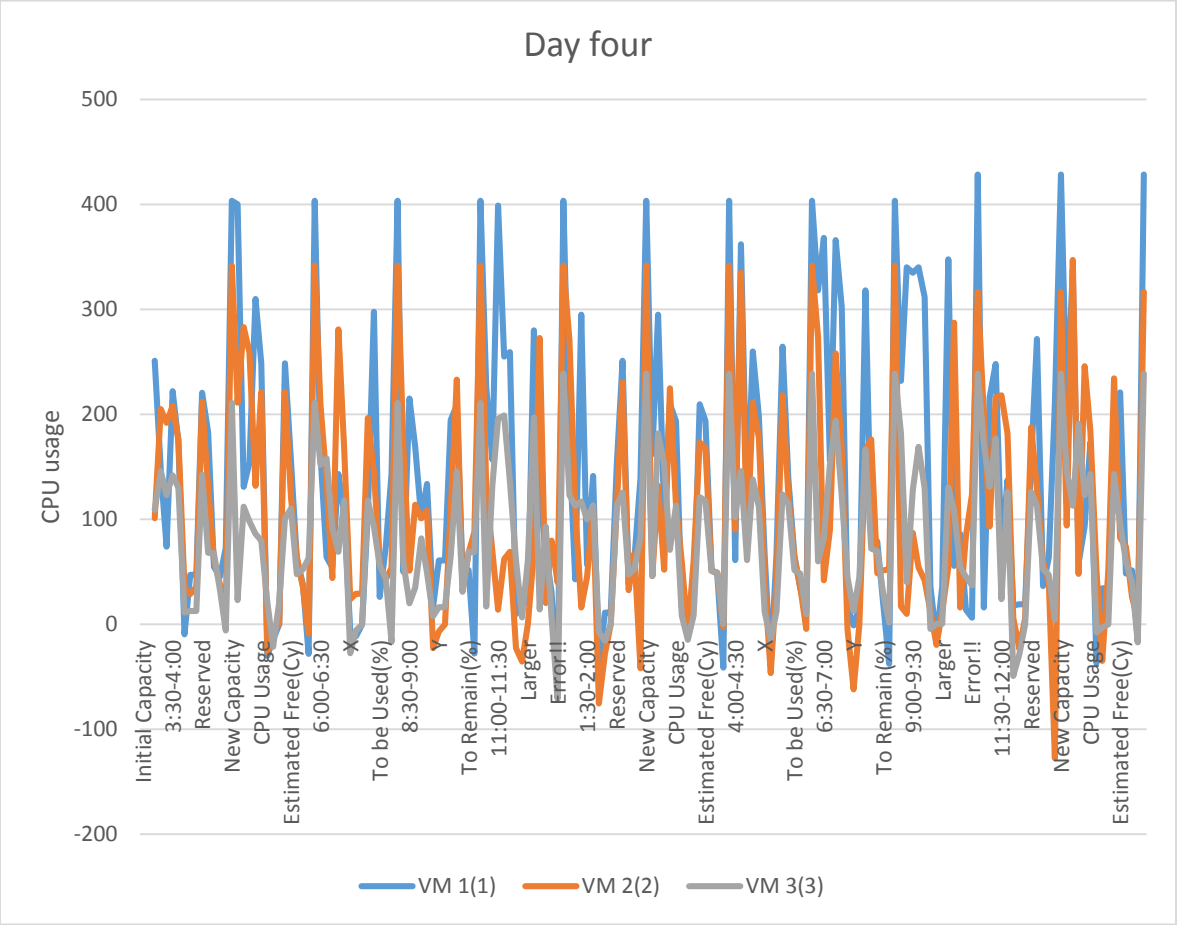

Figure 5-7: day four resource utilization

The graph illustrate that time from 10 to 12 VM3 threshold value is 93.3 %. This show that the scalar reserve resource for next two hours by 13.3% of 210 which previous CPU capacity, it become 238 cycle. VM1 and VM2 at this time is worked with safe state .60% and 20% of CPU usage.403 and 341 cycle .from time 8-10 ,VM1 has reserve resource for net two hours since its threshold value become 86.6 % of 403 cycle .this indicate resource is scale in 6.6% of 403 cycle

for next two hour 428 cycle . The other virtual machine is continue as their capacity, VM2 316 and VM3 238. The total error prediction error for VM1, VM2 and VM3 has 0.305 %, 0.155 %, 0.13% respectively

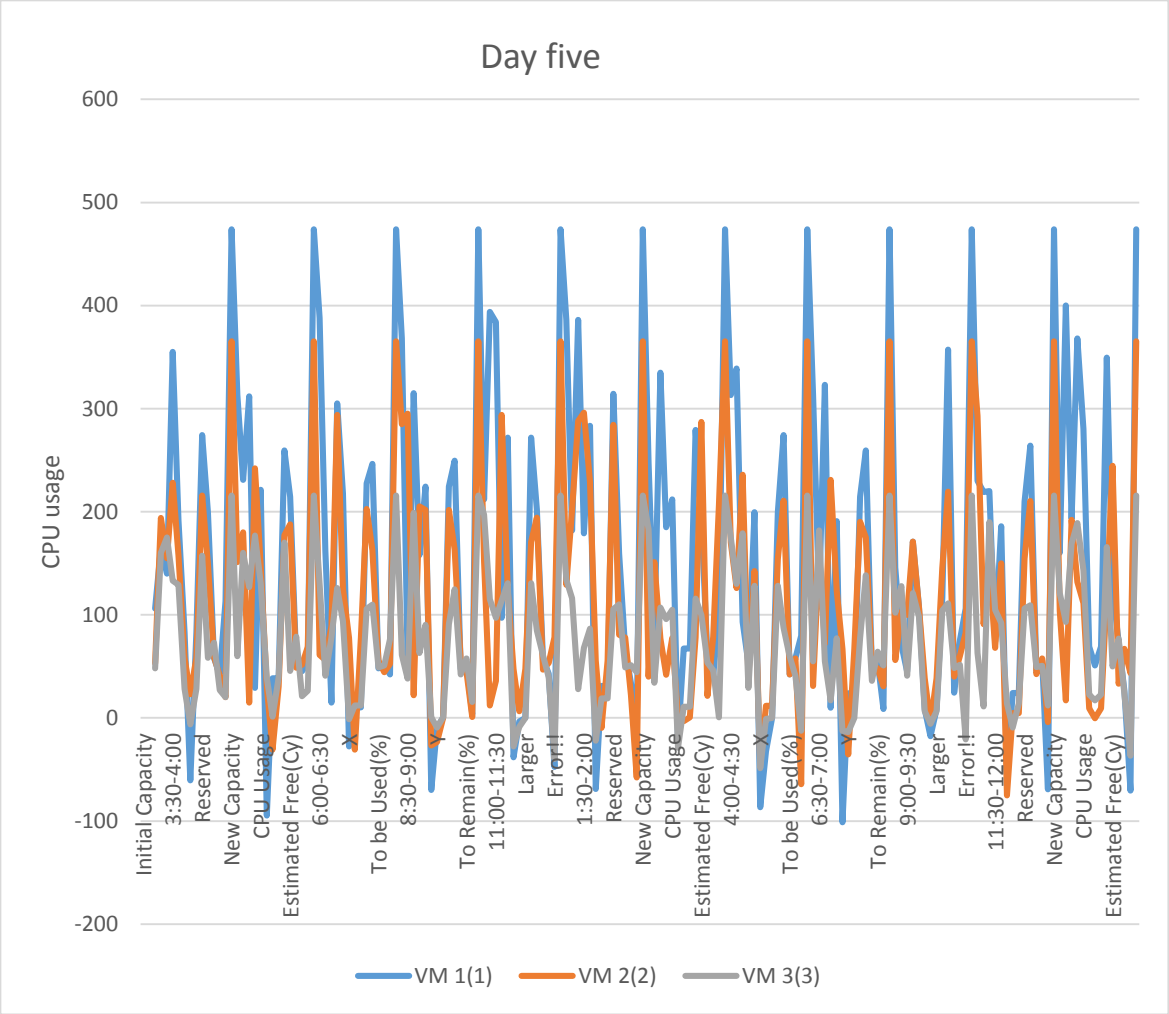

Figure 5-8: day five resource utilization

Form the above graph we understand initially the VM capacity has 433 , 375 ,207 cycle for VM1, VM2 and VM3 respectively ,on this day show that no virtual machines is ask above its capacity ,the whole day they finish with the resource that they have at the beginning . The prediction errors has 0.162%, 0.265% and 0.173% for VM1, VM2 and VM3 respectively .

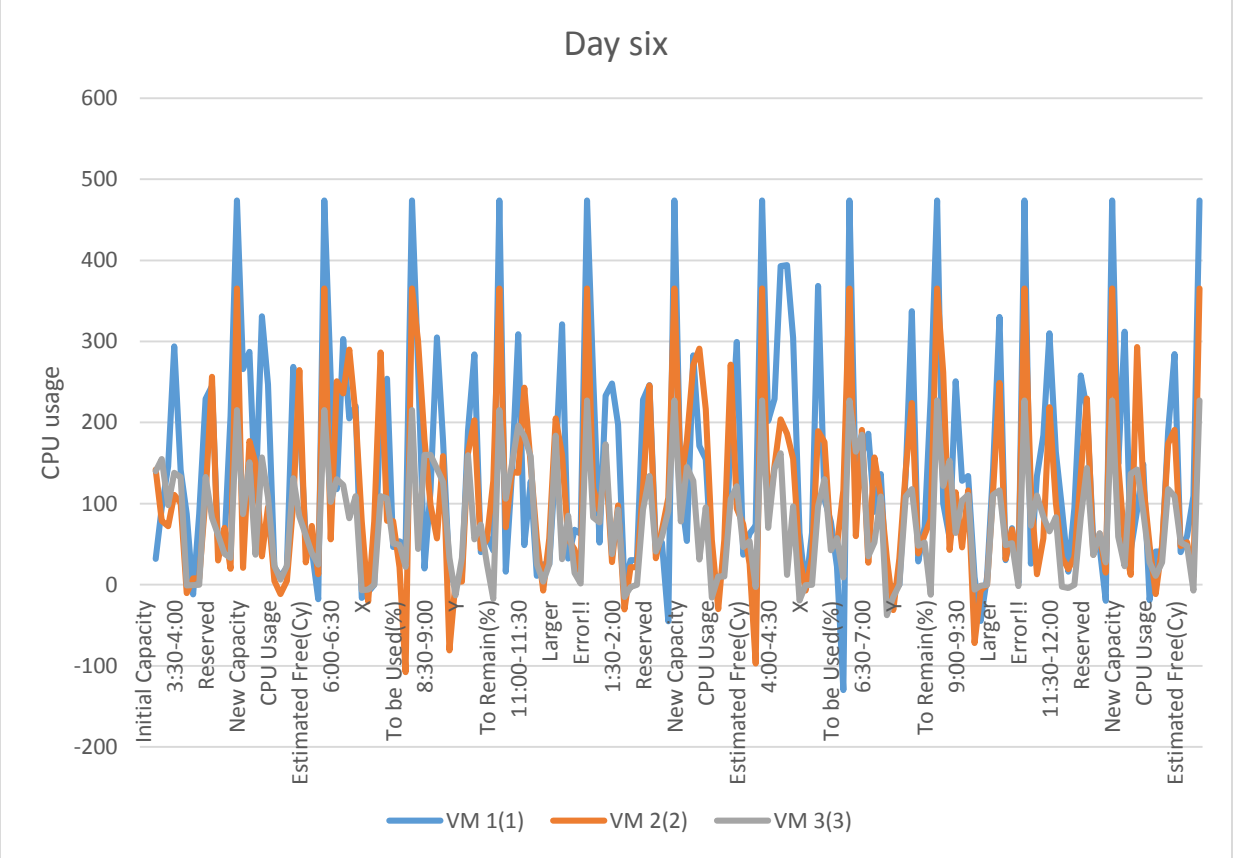

Figure 5-9: day six resource utilization

On day six, as the graph show on the time 12-2, VM1 have more load so resource need to be scaled for next two hour, thresh hold value is 81.7% of 449 cycle .reserve coefficient value is 85 cycle from the current CPU usage 281.5 .it increase based on coefficient rate value on hour since it is greater than the value estimated on day. For the next 2 hour, 1.7% of 449 cycle is reserved, based on that value, the next resource scaled in to 457 cycle, total prediction error is 0.096 %, 0.100 % and 0.143% of VM1, VM2 and VM3

On day seven there is not more load to scale in the resource on virtual machines , all the Three VM have finish their job with the same CPU cycle as the begin their job , 457, 402, 214 of VM1, VM2 and VM3 respectively . In this day, total prediction error is 0.326 %, 0.277 % and 0.037% of VM1, VM2 and VM3 respectively.

For all these day the total prediction error is, 0.003 % for all virtual machines as sown in the table below

Table 5-1: total prediction error for VM

Duration	VM1	VM2	VM3
Day One	0.521	0.142	0.171
Day Two	0.131	0.034	0.104
Day Three	0.369	0.305	0.148
Day Four	0.305	0.155	0.13
Day Five	0.162	0.265	0.173
Day Six	0.096	0.100	0.143
Day Seven	0.326	0.277	0.037
Total	1.91	1.27	0.96

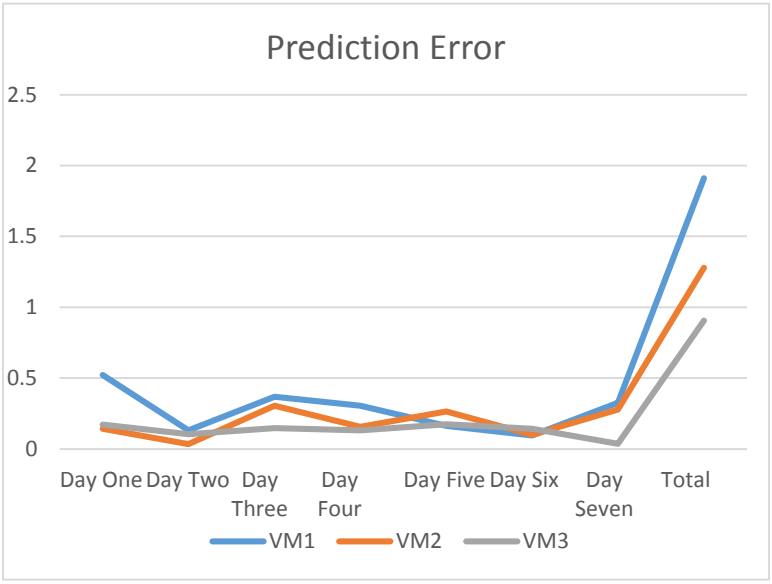

Figure 5-10: prediction error for VM

5.3. Evaluation of analysis

From the result indicate that our algorithm prediction error is less than 2% which is better than the other previous work. Our algorithm predict for the next execution time interval, it reserve extra resource, well know cloud providers are paying system are given with instance and volumes, which they are given this service per hour and month as stated in [51] which shown in table below

Table 5-2: comparison of cloud provider cost

Amazon AWS	Microsoft Azure	Google Compute Engine	Rackspace	IBM Smart Cloud	HP Cloud
Virtual CPUs	1-8	1-8	1-8	1-8	1-8
Memory	613MB - 68.4GB	1.7GB - 14GB	3.7GB - 52GB	512MB - 30GB	1GB - 32GB
Cost/HR	Free - $4.60	$0.02 - $2.04	$0.145 - $1.375	$0.022 - $1.20 (Linux)	$0.035 - $1.12 (Linux)
Storage Costs	$0.095 GB/mo (S3)	$0.095	$0.10 GB/mo	$0.15 GB/month	$0.10 GB/mo

As we try to explain that for configuration new virtual machine, provider takes horizontally scaling scheme. At this time it takes 2-5 minute to configure the newly created virtual machines, but if they used vertical scaling scheme it takes 5 sec to configure the virtual machine. Let's take our analysis result, from our experiment we had make scaling for each virtual machines illustrate in Table 5-3. That VM 1 in this day tries to scale 5 times, when we use horizontal scaling it

Table 5-3: VM scaling time with date

day	virtual machine	Time					
		2 to 4	4 to 6	6 to 8	8 to 10	10 to 12	12 to 2
1	vm1						
	vm2						
	vm3						
2	vm1						
	vm2						
	Vm3						
3	vm1						
	vm2						
	vm3						
4	vm1						
	vm2						
	vm3						
6	vm1						
	vm2						
	Vm3						

From the above table we see that VM 1, VM2 and VM3 try to scale 5, 3 and 4 times in a week respectively. If we where use horizontal scaling with maximum time to configure VM 1 will be create other 5 Virtual machine, VM2 will also create other 3 virtual machine and las VM4 will create 4 virtual machine, to configure all this virtual machines it takes its own time, lets we use maximum of 5 minute configuration time on each of cloud provider and also 5 sec if we use vertical scaling for each of cloud provider as shown in the table 3. And also in table two for our estimate cost calculation the hourly paying amount changed to minute and second for each cloud providers

Table 5-4: cloud providers cost with min and sec

Amazon AWS	Microsoft Azure	Google Compute Engine	Rackspace	IBM Smart Cloud	HP Cloud
Cost/HR	4.6	2.04	1.375	1.2	1.12
Cost/mint	0.077	0.034	0.023	0.02	0.018
Cost/sec	0.00128	0.00056	0.00038	0.000333	0.0003

Table 5-5: comparison on VM cost with cloud providers

Name of VM	Microsoft Azure		Google Compute Engine		Rackspace		IBM Smart Cloud		HP Cloud	
	horizontal	vertical	Horizontal	vertical	Horizontal	vertical	horizontal	vertical	Horizontal	Vertical
VM1	1.75	0.032	0.85	0.014	0.575	0.0095	0.5	0.008325	0.45	0.0075
VM2	1.155	0.0192	0.51	0.0084	0.345	0.0057	0.3	0.0049	0.27	0.0045
VM3	1.54	0.0256	0.68	0.0112	0.46	0.0076	0.4	0.0066	0.36	0.006

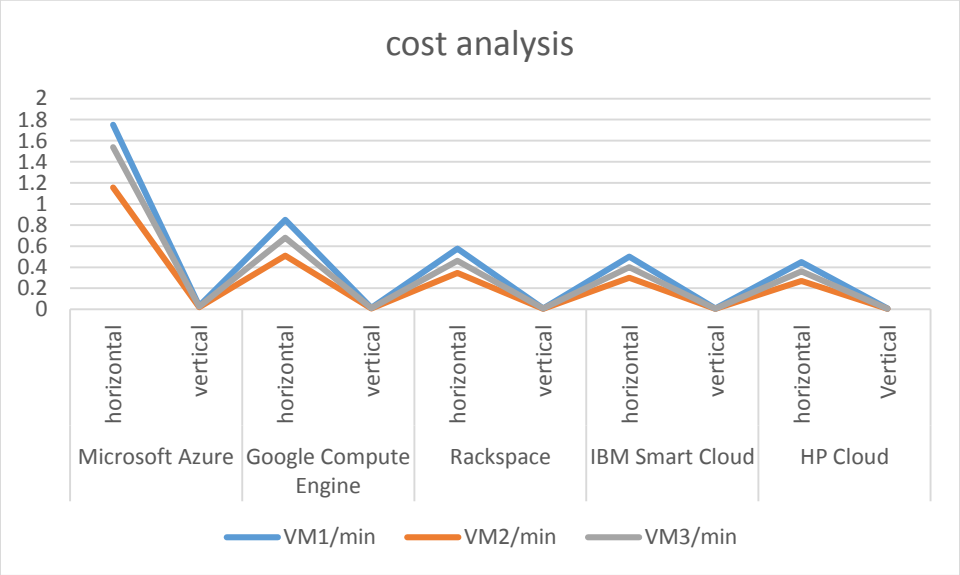

Figure 5-11: cost analysis for 3 VM

The graph clearly show that horizontal scaling has high cost than vertical scaling for different cloud providers , totally this result is based on week analysis , but what is ones cloud customer use amazon with this 3 VM , annual and monthly extra cost for configuration virtual machine is describe in table 5-6

Table 5-6: monthly and yearly cost analysis for VM

Name of VM	Microsoft Azure					
	horizontal	vertical	monthly cost		yearly cost	
			horizontal	Vertically	horizontal	vertically
VM1	1.75	0.032	7	0.128	91	1.664
VM2	1.155	0.0192	4.62	0.0768	60.06	0.9984
VM3	1.54	0.0256	6.16	0.1024	80.08	1.3312

Our algorithm is clearly show that it eliminate this extravagancy , sine the prediction done every next two hour reserve extra resource for the next two hour before the resource demand is high , compare with others work . So from this result customers are not forced to pay other extra money to scale in resource.

Our algorithm use equal priority principle to all VM run in hypervisor, when load above specific threshold, the scaler reserve automatically the resource for the next some two hours. Which is not in other work scaling resource based on priority VM which may cause resource starvation,

CHAPTER SIX: CONCLUSION AND FUTURE WORK

Cloud computing is currently widely used technology that using virtualization technology offering different resources in scalable manner. Two players in cloud computing environments, cloud providers and cloud users, having different goals providers want to maximize their service by achieving high resource utilization, while users want to minimize expenses while meeting their performance requirements. Cloud users pay what the resource they used and also get the resource based on demand .when VM application load increase, cloud resource must scaled up or scaled out to finish the specific task within specific SLA. cur runt cloud platforms are prefers scaled out the resource for generating more money than scaled up the resource, as we try to explain the difference between the scaled up and scale out resource, to configure the virtual machine on these two scheme it have time difference, so the user can pay more on scaled out resource but when resource is scaled up it take an advantage on reducing the operational cost and time taking to configure the virtual machines. We have proposed a clear definition of the scaled out problem, and explained the different experimental platforms used to test scaled up algorithms, together with the different input workloads and evaluation benchmarks available. This chapter states our thesis final out put that fulfilled with the initial aims which is set earlier and also suggest the direction on which related area should follow for comer researcher.

6.1. Conclusions

The goal of this thesis was to design, implement, demonstrate, and evaluation a vertical scalable cloud architecture on some sample web applications This work try to show how to resource vertically scaled up for virtual machines and how the cost be minimize with related to horizontal scaling scheme., especially one which is used by people in a local area , there is going to be a fluctuation of users throughout the day and there is no reason to allocate more or less resources than actually needed hence we focused on how cloud be resource scaled vertically scalable the resource rather than to create other virtual machines and making people for extravagancy ,the architecture provide these solution without weight further time and more costs

The thesis show the importance and usefulness based on metrics defined in cloud that how it could improve the neatness of vertical scaling decisions. We are doing different and several measurements with various setups. The results of performance monitoring were analyzed and used for further improving scaling behavior. We also created a few complex graphs for displaying

scaling events on the same diagram, which help visualize the underlying causalities between metric values and scaling actions while giving a notion about time proportions.

We achieved our aims with vertical scalable proposed architecture, We have demonstrated different guidelines and techniques in order to scale up the resource, we use the techniques hot plunging , we also used scaled up based on threshold value and based on historic data , when the CPU load is come above 80% , the vertical scaling is immediately begin to scale by increasing the difference of actual value with thresh hold value based on the reserved coefficient values , this is done in all time VMs run in hypervisor . if the VM is created for the first time and no resource static information in stack, they use only thresh hold value and scale based on caps value assigned by administrator for the first virtual machine applications which is run on that hypervisor. In our vertical scalable, prediction error has less than 2% Which have better than other proposed solutions Furthermore, this vertical scalability approach also illustrated that it only utilizes resources when needed, avoiding overutilization and underutilization for cloud providers and as a result clearly reduces extravagancy cost for the user. Our findings and cost analysis also clearly shows that can potentially save up to approximate for VM1 in amazon $7 daily and $91 yearly with horizontal scaling scheme a. on a vertically scaling scheme also save up approximately $ 0.128 monthly and $0.66 yearly ,

During this thesis, we have learned various cloud architectures, vertical scaling and dynamic resource allocation mechanisms, and their implementations. The thesis also gave use very useful experience in working with different cloud environment like OpenStack, amazon ,cloud sigma and so on which have their own differences to create instances on this cloud platform .as we demonstrated our experiment on the OpenStack and deploy applications using simple java web applications. Especially when we configure XEN hypervisor and OpenStack cloud plat form we were get a great knowledge , how could this virtual machine monitor and cloud platform management software should be work . This thesis proved to be a very good experience for us. Apart from that, we are confident that this thesis will help us drive our own future career towards cloud technology, as cloud computing is today a very hot topic for the future IT solutions. And also for the reader may get a hands-on experience and a deeper understanding of the case studies, configuration files of scaling and the cloud environment, and step-by-step guides of deployment and trial of vertical scaling.

6.2. Future Work

Due to the limited time and resources during the course of this thesis, it was not possible to perform all the tasks defined what intend in our objective. Moreover, the observations and findings during the course of this thesis suggest some areas for further research. We plan to carry out some of these tasks in the near future. However, other students and researchers may also want to explore one or more of these topics.

- For the proposed framework to be adapted by the cloud providers, a feasible pricing model needs to be developed. For example, in the current system, cloud users are charged fixed bundle instance and hourly billing system
    - These cloud provider are use fixed bundle instance which providers rent their resource within fixed resource number by categorizing with different level (large, medium, small and tiny) in this fixing instance are let user need only 5 CPU cycle, the provider may have not such level that satisfying users CPU cycle then the user is forced to rent more CPU cycle that is pay for un used CPU cycle which lead over provisioning.
    - The other is cloud provider rent their infrastructure with hourly billing system this leads also for more extravagancy that incase user needs these resource for 10 or 20 minute they will pay for which minute they are not used. But they have to pay for the time only they had used

REFERENCES

[1] D. C. Marinescu, Cloud Computing Theory and Practice Cloud Computing Theory and Practice, Nework: Elsevier In, 2013 , p. 132.

[2] Marrian Turowski , Alexander Lenk, "Vertical Scaling Capability of OpenStack Survey of Guest Operating Systems, Hypervisors, and the Cloud Management Platform Marian," *FZI Forschungszentrum Informatik*, p. p.4, 2014.

[3] V.Vinothina, Dr.R.Sridaran, Dr.PadmavathiGanapathi, "A Survey on Resource Allocation Strategies in Cloud Computing," *International Journal of Advanced Computer Science and Applications*, vol. volume 3, p. No. 6, 2012.

[4] D. Bellenger, J. Bertram, A. Budina, A. Koschel, B. Pf¨ander, C. Serowy, I. Astrova, S. G. Grivas, and M. Schaaf, "Scaling in cloud environments," in *in Proceedings of the 15th WSEAS international conference on Computers* , Stevens Point, Wisconsin, USA, 2011.

[5] Marian Turowski , Alexander Lenk, "Vertical Scaling Capability of OpenStack Survey of Guest Operating Systems, Hypervisors, and the Cloud Management Platform," *turowskivertical*, p. p.8, 2014.

[6] Andrew J. Younge, Robert Henschel, James T. Brown, Gregor von Laszewski, Judy Qiu, Geoffrey C., "Analysis of Virtualization Technologies for High Performance Computing Environments," *Fox Pervasive Technology Institute.*

[7] "https://aws.amazon.com/," [Online].

[8] P. Mell and T. Grance, "The nist definition of cloud computing," National Institute of Standards and Technology," Vols. vol, 53, p. p.50, 2009.

[9] R. Buyya, C. S. Yeo and S. Venugopal,, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," in High Performance Computing and Communications,HPCC .," in *10th IEEE International Conference*, 2008.

[10] "Cisco Systems, "Cloud Computing - Overview," 12 08 2015. [Online].

[11] J. Hurwitz, R. Bloor, M. Kaufman, and F. Halper, , "What Is Cloud Computing," 21 06 2015. [Online].

[12] Foster, I., Zhao, Y., Raicu, I. & Lu, S, "Cloud computing and grid computing 360-degree compared.," in *IEEE Grid Computing Environment*, 2008.

[13] Keith Jeff ery ,Burkhard Neidecker-Lutz, The future of cloud computing ,oportunuty for europian cloud computing beyond, 2010, p. p.12.

[14] W. Voorsluys, J. Broberg, and R. Buyya, Introduction to Cloud ComputingR. Buyya, J. Broberg, and A. Goscinski, Eds. ,,, John Wiley & Sons, Inc., 2011, p. pp. 1– 41.

[15] "Amazonec2," 2015. [Online].

[16] "Google App Engine," 12 10 2015. [Online].

[17] " Google App Engine," 2015. [Online].

[18] "https://www. aws.amazon.com/ elasticbeanstalk," 2015. [Online].

[19] "Windows Azure," 2015. [Online].

[20] "Defining Cloud Computing's Key Characteristics, Deployment and Delivery Types," 2015. [Online].

[21] Lenar Yazdanov and Christof Fetzer., "Vertical scaling for prioritized vms provisioning," in *International Conference on Cloud and Green Computing. IEEE Computer Society*, 2012.

[22] T. Harris, Cloud Computing Services – A comparison, 2014.

[23] Jinesh Varia,Sajee Mathew, "Overview of Amazon Web Services," 2014.

[24] "Amazon Machine Images," 2015. [Online].

[25] May Al-Roomi, Shaikha Al-Ebrahim, Sabika Buqrais and Imtiaz Ahmad, "Cloud Computing Pricing Models: A Survey," *International Journal of Grid and Distributed Computing*, vol. 06, 2013.

[26] S. Goyal, "A COMPARATIVE STUDY OF CLOUD COMPUTING SERVICE PROVIDERS ,," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 2, p. 2, 02 2012.

[27] "Amazon. Amazon Elastic Compute Cloud (Amazon EC2)," 25 09 2015. [Online].

[28] "https://www.windowsazure.com," 05 2015. [Online].

[29] J. Mäenpää, "Cloud computing with the Azure platform,," 2009.

[30] "MICROSOFT, SQL Azure.," 28 05 2015. [Online].

[31] D. Chappel, "Introducing the Windows Azure Platform," 05 2015. [Online].

[32] "software," 2015. [Online].

[33] Openstack installation guide for Ubuntu 14.04, 2015, p. 44.

[34] "Eucalyptus 3.4.2 User Guide," Eucalyptus Systems, 2014.

[35] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov, "The Eucalyptus Open-source Cloud-computing System".

[36] Eucalyptus Cloud Computing Platform Administrator's Guide Enterprise Edition 2.0, 2010, p. p. 16.

[37] M. Rosenblum, "The reincarnation of virtual machines," 08 2004.

[38] R. M. Sharma, "The Impact of Virtualization in Cloud Computing," *International Journal of Recent Development in Engineering and Technology,* vol. 3, no. 1, 07 2014.

[39] H. Saltzer and M. F. Kaashoek, Principles of Computer System Design., Nework: Morgan Kaufamnn , 2009.

[40] Kevin Lawton, Bryce Denney, N. David Guarneri, Volker Ruppert, Christophe Bothamy, and Michael Calabrese, "Bochs x86 pc emulator users manual," 2003. [Online].

[41] A. S. Tanenbaum, MODERN OPERATING SYSTEMS THIRD EDITION, Pearson Education, Inc, 2009.

[42] Durairaj. M, kannan.p , " A study on virtualization techniques and challenges in cloud computing," *international journal of scientific & technology research* , vol. 3, no. 11, 11 2014.

[43] "Wine user guide," 2015. [Online].

[44] Bo Yin, Ying Wang, LuomingMeng, XuesongQiu, "A Multi-dimensional Resource Allocation Algorithm in Cloud Computing," Beijing University of Posts and Telecommunications, Beijing , 2012.

[45] Zhen Xiao, Weijia Song, and Qi Chen, "Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment," *IEEE Transactions on parallel and distributed systems,,* vol. 24, no. 6, 06 2013.

[46] Jiayin Li, MeikangQiu, Jian-Wei Niu, Yu Chen, Zhong Ming,, "Adaptive Resource Allocation for Pre-empt able Jobs in Cloud Systems," in *10th International Conference on Intelligent System Design and Application,*, 2011.

[47] Lenar Yazdanov and Christof Fetzer., "Vscaler :autonomic virtual machine scaling," 2013.

[48] Dutta S., Gera S., Verma A., and Viswanathan B, "SmartScale: Automatic Application Scaling in Enterprise Clouds.," in *2012 IEEE 5th International Conference on In Cloud Computing* .

[49] "https:/www.Google Analytics_ Hour of Day & Day of Week Reports _ Hallam Internet.html," 2015. [Online].

[50] Dawoud, W., Takouna, I., and Meinel, C., "Elastic Virtual Machine for Fine-Grained Cloud Resource Provisioning.," in *Global Trends in Computing and Communication Systems*, 2012.

[51] "http://www.tomsitpro.com/articles/cloud-computing-iaas-providers-comparison,2-583-3.html," 07 02 2015. [Online].

APPENDICES

APPENDICES A Configuring OpenStack

Prerequisites Linux & Network

Install a couple of packages to bootstrap configuration:

apt-get install -y git sudo || yum install -y git sudo

Network Configuration

The first iteration of the lab uses OpenStack's FlatDHCP network controller so only a single network will be required. It should be on its own subnet without DHCP; the host IPs and floating IP pool(s) will come out of this block. This example uses the following:

Gateway: 192.168.42.1

Physical nodes: 192.168.42.11-192.168.42.99

Floating IPs: 192.168.42.128-192.168.42.254

Configure each node with a static IP. For Ubuntu edit /etc/network/interfaces:

```
auto eth0
iface eth0 inet static
    address 192.168.42.11
    netmask 255.255.255.0
    gateway 192.168.42.1
```

OpenStack runs as a non-root user that has sudo access to root. There is nothing special about the name, we'll use stack here. Every node must use the same name and preferably uid. If you created a user during the OS install you can use it and give it sudo privileges below. Otherwise create the stack user:

groupadd stack

useradd -g stack -s /bin/bash -d /opt/stack -m stack

This user will be making many changes to your system during installation and operation so it needs to have sudo privileges to root without a password:

echo "stack ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers

From here on use the stack user. **Logout** and **login** as the stack user.

Set Up Ssh

Set up the stack user on each node with an ssh key for access:

mkdir ~/.ssh; chmod 700 ~/.ssh

echo "ssh-rsa

AAAAB3NzaC1yc2EAAAADAQABAAABAQCyYjfgyPazTvGpd8OaAvtU2utL8W6gWC4Jd

RS1J95GhNNfQd657yO6s1AH5KYQWktcE6FO/xNUC2reEXSGC7ezy+sGO1kj9Limv5vrvN

HvF1+wts0Cmyx61D2nQw35/Qz8BvpdJANL7VwP/cFI/p3yhvx2lsnjFE3hN8xRB2LtLUopUS

VdBwACOVUmH2G+2BWMJDjVINd2DPqRIA4Zhy09KJ3O1Joabr0XpQL0yt/I9x8BVHdAx

6l9U0tMg9dj5+tAjZvMAFfye3PJcYwwsfJoFxC8w/SLtqlFX7Ehw++8RtvomvuipLdmWCy+T9

hIkl+gHYE4cS3OIqXH7f49jdJf jesse@spacey.local" > ~/.ssh/authorized_keys

Download DevStack

Grab the latest version of DevStack:

git clone https://git.openstack.org/openstack-dev/devstack

cd devstack

Up to this point all of the steps apply to each node in the cluster. From here on there are some differences between the cluster controller (aka ‘head node’) and the compute nodes.

Configure Cluster Controller

The cluster controller runs all OpenStack services. Configure the cluster controller's DevStack in`local.conf`:

[[local|localrc]]

HOST_IP=192.168.42.11

FLAT_INTERFACE=eth0

FIXED_RANGE=10.4.128.0/20

FIXED_NETWORK_SIZE=4096

FLOATING_RANGE=192.168.42.128/25

MULTI_HOST=1

LOGFILE=/opt/stack/logs/stack.sh.log

ADMIN_PASSWORD=labstack

DATABASE_PASSWORD=supersecret

RABBIT_PASSWORD=supersecrete

SERVICE_PASSWORD=supersecrete

In the multi-node configuration the first 10 or so IPs in the private subnet are usually reserved.

Add this to `local.sh` to have it run after every `stack.sh` run:

for i in `seq 2 10`; do /opt/stack/nova/bin/nova-manage fixed reserve 10.4.128.$i; done

Fire up OpenStack:

./stack.sh

Configure Compute Nodes

The compute nodes only run the OpenStack worker services. For additional machines, create a`local.conf` with:

[[local|localrc]]

HOST_IP=192.168.42.12 # change this per compute node

FLAT_INTERFACE=eth0

FIXED_RANGE=10.4.128.0/20

FIXED_NETWORK_SIZE=4096

FLOATING_RANGE=192.168.42.128/25

MULTI_HOST=1

LOGFILE=/opt/stack/logs/stack.sh.log

ADMIN_PASSWORD=labstack

DATABASE_PASSWORD=supersecret

RABBIT_PASSWORD=supersecrete

SERVICE_PASSWORD=supersecrete

DATABASE_TYPE=mysql

SERVICE_HOST=192.168.42.11

MYSQL_HOST=$SERVICE_HOST

RABBIT_HOST=$SERVICE_HOST

GLANCE_HOSTPORT=$SERVICE_HOST:9292

ENABLED_SERVICES=n-cpu,n-net,n-api-meta,c-vol

NOVA_VNC_ENABLED=True

NOVNCPROXY_URL="http://$SERVICE_HOST:6080/vnc_auto.html"

VNCSERVER_LISTEN=$HOST_IP

VNCSERVER_PROXYCLIENT_ADDRESS=$VNCSERVER_LISTEN

Note: the n-api-meta service is a version of the api server that only serves the metadata service. It's needed because the computes created won't have a routing path to the metadata service on the controller.

Fire up OpenStack:

./stack.sh

A stream of activity ensues. When complete you will see a summary of stack.sh's work, including the relevant URLs, accounts and passwords to poke at your shiny new OpenStack. The most recent log file is available in stack.sh.log.

Cleaning Up After DevStack

Shutting down OpenStack is now as simple as running the included unstack.sh script:

./unstack.sh

Swift

Swift, OpenStack Object Storage, requires a significant amount of resources and is disabled by default in DevStack. The support in DevStack is geared toward a minimal installation but can be used for testing. To implement a true multi-node test of swift, additional steps will be required. Enabling it is as simple as enabling the swift service in local.conf:

enable_service s-proxy s-object s-container s-account

Swift, OpenStack Object Storage, will put its data files in SWIFT_DATA_DIR (default/opt/stack/data/swift). The size of the data 'partition' created (really a loop-mounted file) is set bySWIFT_LOOPBACK_DISK_SIZE. The Swift config files are located in SWIFT_CONF_DIR (default /etc/swift). All of these settings can be overridden in (wait for it...) local.conf.

Volumes

DevStack will automatically use an existing LVM volume group named stack-volumes to store cloud-created volumes. If stack-volumes doesn't exist, DevStack will set up a 10Gb loop-mounted file to contain it. This obviously limits the number and size of volumes that can be

created inside OpenStack. The size can be overridden by setting VOLUME_BACKING_FILE_SIZE in local.conf.

stack-volumes can be pre-created on any physical volume supported by Linux's LVM. The name of the volume group can be changed by setting VOLUME_GROUP in localrc. stack.sh deletes all logical volumes in VOLUME_GROUP that begin with VOLUME_NAME_PREFIX as part of cleaning up from previous runs. It is recommended to not use the root volume group as VOLUME_GROUP.

The details of creating the volume group depends on the server hardware involved but looks something like this:

```
pvcreate /dev/sdc
vgcreate stack-volumes /dev/sdc
```

Set MySQL Password

If you forgot to set the root password you can do this:

```
mysqladmin -u root -pnova password 'supersecret'
```