National College of Ireland

Management and Scheduling of Resources along with their Optimal Usage

MSc Research Project
Cloud Computing

Akanksha Dadhich
x17131618

School of Computing
National College of Ireland

Supervisor: Divyaa Manimaran Elango

National College of Ireland
Project Submission Sheet – 2017/2018
School of Computing

Student Name:	Akanksha Dadhich
Student ID:	x17131618
Programme:	Cloud Computing
Year:	2016
Module:	MSc Research Project
Lecturer:	Divyaa Manimaran Elango
Submission Due Date:	20/12/2018
Project Title:	Management and Scheduling of Resources along with their Optimal Usage
Word Count:	6424

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:	
Date:	27th January 2019

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
3. Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Management and Scheduling of Resources along with their Optimal Usage

Akanksha Dadhich
x17131618
MSc Research Project in Cloud Computing

27th January 2019

Abstract

Cloud computing is the branch of technology that offers the users to utilize various computational, networking, storage, software and hardware resources on pay as you go basis. Several organizations are migrating to cloud as it is a cost-effective solution. This paper aims to solve the problem of resource leakage (memory, disk space, network channels, virtual cores, etc.) due to the generation of orphan VMs. The optimum usage of resources is often restricted due to the creation of orphan VMs. Hence, the demands of users are not fulfilled. This is an emerging problem in various cloud platforms such as OpenStack, VMWare, etc. The orphan VMs decreases the availability of resources and thus results in certain delays or deadlocks, interruption of service, degraded system performance, depletion of the pool of shared resources, etc. The objective of this paper is to overcome the problems caused by orphan VMs by implementing an appropriate solution for the end to end termination of VMs. To conduct the research, OpenStack cloud has taken into consideration.

Keywords: optimization, orphan VMs, availability, cloud computing.

- Motivation:

Resource optimization is becoming a rising issue in the world of computing which can lead to decreased availability of resources along with service interruption. The major reasons responsible behind this problem can be resource leakage, various attacks, data corruption, etc. which further results in VM leakage and generation of orphan VMs. Orphan VMs sound like a very common and unnecessary glitch, that's why it is often ignored and not taken seriously. But coming to the ill effects it can create, I found that, it can not only degrade the availability of resources but also result in system failures. By researching a little more about this, I evaluated that, a strong solution to eradicate this problem has not been implemented up to now. Thus, I decided to carry on my research on this problem to propose and implement an appropriate solution to eradicate the problem of orphan VMs along with the proper management and scheduling of resources.

Contents

1 Introduction

- Research Question:

  How Resource Management and Scheduling can be improved using ansible script for end to end termination of Orphan VMs?

Nowadays, migrating to cloud platform is trending pretty much because of the attractive and beneficial features provided by various cloud service providers Paper (2014). Cloud computing is the branch of computer science that ensures a successful and bright future of the modern world of computing. With the use of cloud computing, there is seen approximately 10 times rise of opportunities for the organizations. It is a service which offers a wide range of resources which can be used for automation, standardization, optimization, etc. Along with this, the users are also provided with a shared pool of resources for examples, computing resources, storage resources, networking resources, etc. By looking at the following figure, the concept of cloud computing and the services provided can be explained more precisely:

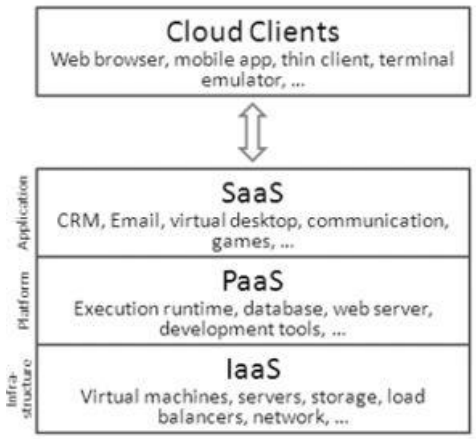

Figure 1: Services Offered by Cloud Computing

The above Figure 1 depicts that cloud computing possess three service models namely IaaS (Infrastructure-as-a-Service), PaaS (Platform-as-a-Service) and SaaS (Software-as-a-Service). IaaS (Infrastructure-as-a-Service) is used for accessing, networking services, monitoring and managing the remote data centres (compute, networking, storage, etc.) Example: Amazon EC2. In PaaS (Platform-as-a-Service), the user is able to create one's own software applications. Example: Windows Azure. Coming to SaaS (Software-as-a-Service), here all the applications are available to the users remotely. Example: GoogleApps Barkat et al. (2014). Other than the service models, cloud computing also possess four deployment models i.e. Private Cloud, Public Cloud, Hybrid Cloud and Community Cloud.

This paper is focused to the detection, and elimination of orphan VMs in modern data-centers. In order to the understand the concept of orphan VMs, it is first required to go through the problem of resource optimization and its consequences Dabrowski and Mills (2011). Optimization of resources has become a leading problem in the present

era of technology and computing. It further leads to service interruption, decreased availability of resources, etc. The major causes behind the issue of resource optimization can be corruption of data, resource leakage, several attacks, etc. all these can result into VM leakage along with the generation of orphan VMs. Generation of orphan VMs is the rising issue in modern data-centers. In simple words, orphan VMs are the error VMs which unnecessarily occupy the vacant spaces in the database and hence, no further VMs can be generated when required by the user. Thus, there is a serious degradation in the availability of resources due to generation of orphan VMs HERLIHY et al. (1989).

The further sections of this paper describes the complete process of generation of orphan VMs along with proposing a solution to detect and eliminate them. This issue is often encountered in various cloud platforms such as VMWare, OpenStack, AWS, Microsoft Azure, etc. Here, in this research OpenStack Cloud is chosen to run and evaluate the solution. It was launched by Rack-space Hosting and NASA in the year 2010. OpenStack has four characteristics i.e. open source, open development, open design and open community. Following points are taken into consideration before choosing OpenStack:

- OpenStack is an open cloud platform and here several computing resources can be provided to the users.
- It is a free cloud platform and has minimal Capex, Opex and developing expenses as compared to other cloud platforms. Hence, it is cost effective.
- It is one of the largest middle-ware development platforms.
- The time taken in executing VMs is quite less in OpenStack cloud as compared to other cloud platforms.
- It provides the ability to manage and control the computational, networking and storage resources.
- Additionally, the resources are provided to the users on demand by OpenStack.
- A detailed description of OpenStack and its components is provided in the upcoming sections of the paper.

2 Related Work

2.1 Background Research

In the modern era of computing and technology, a continuous growth in the requirement of data-centers is encountered. The major reasons responsible for this growth is the increasing requirement of computational, networking and storage resources dm Mann (2017). Apart from this, one more reason responsible for the growing demand of data-centers is the cloud platform that offers attractive features to the users such as better performance, higher availability of resources, improved security, etc. Migration to cloud platform not only lowers down the Capex (Capital Expenditure) but also provides cost cutting in the Opex (Operational Expenditure). Along with this, the organizations can also achieve the SLO (Service-Level Objectives) easily after migrating to cloud.

Virtualization technology is widely used nowadays in order to assure 100 percent utilization or high availability of resources. Virtualization technology possess facilities like server consolidation, allocation of virtual machines, etc. resulting in various benefits such as better security, improved percentage utilization of resources, mobility to access the data, cost cutting, optimized resource deployment, etc. Virtualization also has a distributed nature as a result of which management and scheduling of resources is not easy Parikh et al. (2017). Virtual Machines (VMs) play a vital role in proper allocation of resources in modern data-centers Gong et al. (2014). The VM (virtual machine) images represents the computational resources. They are of various types and each VM possess various features such as memory, cost, number of cores, etc. Theng and Hande (2012). The process of generating a VM (virtual machine) is explained in detail in the following sections of the paper.

A research has been conducted by Dabrowski and Mills (2011) addressing VM leakage and orphan VMs. The term orphan here refers to leaked resources (VMs) or error VMs. This research describes the benefits and necessity of implementing an orphan control method. For this purpose, a simple experiment is carried out based on a scenario in which a Trojan attack introduces malicious code modifications into one part of an open source cloud implementation. The comparison of leakage of resources with and without an orphan control method is done. It is suggested to create an orphan control method using Eucalyptus protocol, but it is not implemented in this research.

Other than this, there is one more research HERLIHY et al. (1989) conducted by MAURICE P. HERLIHY and MARTIN S. MC KENDRY based on time-stamp for elimination of orphans. It includes two versions i.e. eager orphan elimination and lazy orphan elimination. Both of the versions further include a set of protocols. This proposal was not proved efficient enough to meet the problems raised due to the generation of orphan VMs.

The research paper guang Ao et al. (2016) suggests that although a number of solutions for optimization of resources has been proposed but most of them are focused on service improvement of resources. Hence, none of them is not efficient enough to deal with the limitations caused due to the generation of orphan VMs and VM leakage.

This research is basically focused on the limitations caused due to VM leakage or generation of orphan resources. The paper explains the problem in detail along with proposing a suitable solution in order to overcome the problem.

2.2 Literature Review

2.2.1 Overview of the Problem

As specified in the previous sections of the paper, the problem addressed by this research is the generation of orphan VMs and its drawbacks which degrades the system performance. Orphan resources or resource leakage can take place due to several reasons such as programming errors, poor designing, corruption of data, malicious attacks, etc. Not only this, in severe cases, it can also result in system breakdown. Since, this research is focused to VM leakage and generation of orphan VMs, it is important to understand the concept of orphan VMs and how they are generated.

A previous study Dabrowski and Mills (2011) states that orphan VMs are generated in open source cloud platforms as a result of Trojan attacks. It means that the when the Trojan code enters the open source cloud, it sweeps away the web messages which further results into degraded system performance, decreased availability of resources and generation of orphan VMs. Orphan VMs unnecessarily occupy the free resources which initially degrades the system performance and availability of resources but can even leads to system failure.

The concept of Orphans is explained more precisely by Jahanshahi et al. (n.d.) and Barbonis (2006). These papers states that whenever a request is sent to the receiver by a system and unfortunately the system fails down, then the respective receiver is termed as orphan. Generation of orphan VMs is the rising issue in modern data-centers. It leads to decreased availability of resources along with service interruption. Various data-centers have been facing the problem of orphan VMs since very long. Given below are the limitations caused due to generation of orphan VMs:

- Degraded availability of resources.
- Leakage of resources (memory, disk space, network channels, virtual cores, etc.)
- Depletion of the pool of shared resources.
- Degraded system performance which can even lead to system failure.
- Orphans waste the system resources and they may observe inconsistent data.
- They consume processor cycles.
- Certain delays and deadlocks are encountered.
- Orphans can even complicate the interactive programs.

The background of the research shows that no solution yet proposed is effective enough to overcome the problems caused due to the generation of orphan VMs. Hence, there is a serious requirement of an appropriate solution which detects if there are any orphan VMs generated in the database and if yes, then immediately eliminates them. Elimination of orphan VMs have several benefits such as:

- Improved availability of resources
- Improved system performance
- Improved resource management
- Problem of resource leakage will be eliminated
- Decreased expenses

2.3 About OpenStack Cloud

The proposed solution will be used by the cloud service provider as this problem is encountered in allotting new resources to the users by the service providers. This solution is basically required by the private clouds. For carrying out this research, OpenStack cloud is taken into consideration. Following points are taken into consideration before choosing OpenStack:

- OpenStack is an open cloud platform and here several computing resources can be provided to the users.
- It is a free cloud platform and has minimal Capex, Opex and developing expenses as compared to other cloud platforms. Hence, it is cost effective.
- It is one of the largest middle-ware development platforms.
- The time taken in executing VMs is quite less in OpenStack cloud as compared to other cloud platforms.
- It provides the ability to manage and control the computational, networking and storage resources.
- Additionally, the resources are provided to the users on demand by OpenStack.

OpenStack cloud works on four core principles i.e.

Open Source: The resources which have been released under Apache 2.0 license can be used for free by the users.

Open Design: It means that new releases are made as per the requirements of development team members.

Open Development: The source code of development process is available to everyone.

Open Community: Users can access OpenStack cloud for free, this forms a community of users.

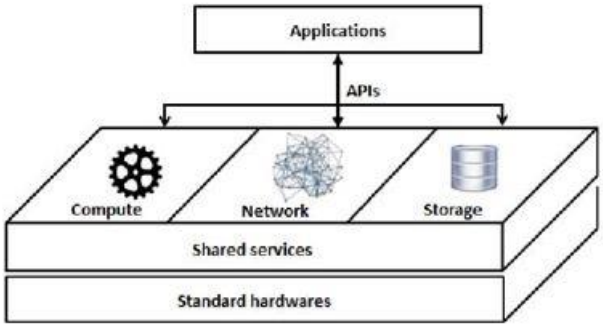

Figure 2: OpenStack Architecture

Above given Figure 2 depicts the architecture of OpenStack cloud. According to Xu and Yuan (2014), OpenStack cloud platform can be built and configured on any physical device such as servers, network devices, disks, etc. A number of virtualization layers can also be built on the OpenStack cloud platform. As per the research article Jain et al. (2016), the architecture of OpenStack cloud comprises of three main components (also termed as three pillars of architecture of OpenStack cloud) namely:

- Compute (Nova)
- Network (Quantum)
- Storage (Swift)

Figure 2 shows that OpenStack is built on standard hardware along with providing several shared services Konoor (n.d.). Despite this, it also consists of the three components which are explained below with detailed description.

i. Compute (Nova): The huge network of VMs (virtual machines) are controlled and managed by this component. Along with this, it also manages the scheduling of VMs (virtual machines) as per the availability of PMs (Physical machines). Nova is the code-name assigned to the compute component. It is also known as the heart of OpenStack cloud. It is a distributed application and is mainly written in Python Teixeira (2014). The architectural diagram of compute is as given below in Figure 3:

The compute (Nova) component controls and manages the complete life-cycle of a VM (virtual machine) right from generation of a VM to the termination of a VM. It is comprised of six sub-components which are stated as follows:

- Nova-api
- Message Queue
- Nova-Compute
- Nova-Network
- Nova-Volume
- Nova-Scheduler

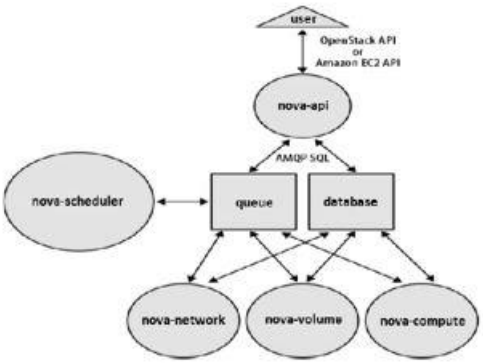

Figure 3: Compute (Nova) Architecture

ii. Network (Quantum): The second component of OpenStack architecture i.e. network component is used to provide networking models, plug-in architectures, APIs, etc. to the users. The users are able to manage workloads and utilize multiple network technologies with the help of network component. It is also known as the key of OpenStack cloud because of its special features such as:

- All the resources and services are accessible here.
- It has the automatic network configuration feature enabled.
- It also supports multi-tier applications.

In order to control the workloads, various load balancing algorithms can be used at network level. Following diagram Figure 4 depicts the architecture of network (quantum) component:

The network (quantum) component of OpenStack cloud consists of four physical data-center networks namely:

a. Management Network
b. Data Network
c. External Network
d. API Network

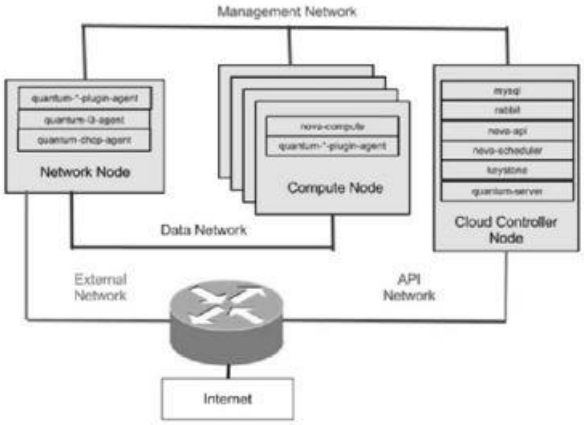

Figure 4: Network (Quantum Architecture)

iii. Storage (Swift): The third component or storage component of OpenStack cloud platform is Swift which is used to control and manage the storage resources. Swift follows the TCP/IP communication due to which problems related to transferring a message are encountered Gugnani et al. (2017). The OpenStack cloud supports two kinds of storage systems namely:

- Object Storage:
  It is an object storage system which is highly scalable.The codename assigned to Object Storage is Swift. The data can be written into several hardware devices in object storage. The main benefits of Object Storage are it possess greater scalability and it is also cost effective Nou et al. (2017) .

- Block Storage:
  It basically establishes connection between the compute instance and block devices. The codename assigned to Block Storage is Cinder. It is better than Object Storage when performance factor is taken into consideration. It possesses eventual consistency by providing synchronization of objects and ability to create a replica of the objects Ruan et al. (n.d.)

3 Methodology

3.1 Proposed Solution

As specified in the introduction section, this research basically deals with the limitations caused due to generation of orphan VMs. The objective of the research is to design a process which detects and eliminates the orphan VMs in OpenStack cloud platform. This will improve both the availability of resources and the system performance. In order to carry out the research, it is first necessary to identify the possible reasons of generation of orphan VMs.

Use Cases

The proposed solution has an urgent requirement in two major sectors:

- Private Banks

  Private Banks often uses private cloud to store their data. VMs are continuously generated and deleted here with the creation and deletion of accounts, adding the new transactions, etc. In such cases, sometimes the orphan VMs can be generated due to unsuccessful deletion or abortion of a transaction. Hence, Private banks can use the orphan control solution to overcome these problems.

- Electronic Commerce Companies:

  During the time of sales for example, Black Friday sale, a very large number of users are expected. Hence, several VMs are continuously generated and deleted. Thus, there is a probability of generation of orphan VMs which often degrades the performance of the website. By using the orphan control solution, the e-commerce companies can overcome this problem.

But before this, it will be beneficial to understand the simple process of creating a VM (virtual machine) or instance on any cloud platform. The given flow chart Figure 5 depicts all the steps involved in generating a VM (virtual machine). As shown in the following chart, in the first step. A request to generate a new virtual machine is made by a user, because of this, authentication of the user takes place. If one is not a registered user, the process is terminated, else availability of resources is checked. If resources are not available, again process is terminated here. If there is an availability of resources, further scheduling steps i.e. scheduling of jobs, image selection, selection of network, etc. are carried out. In this way a VM (virtual machine) is created and the process is stopped.

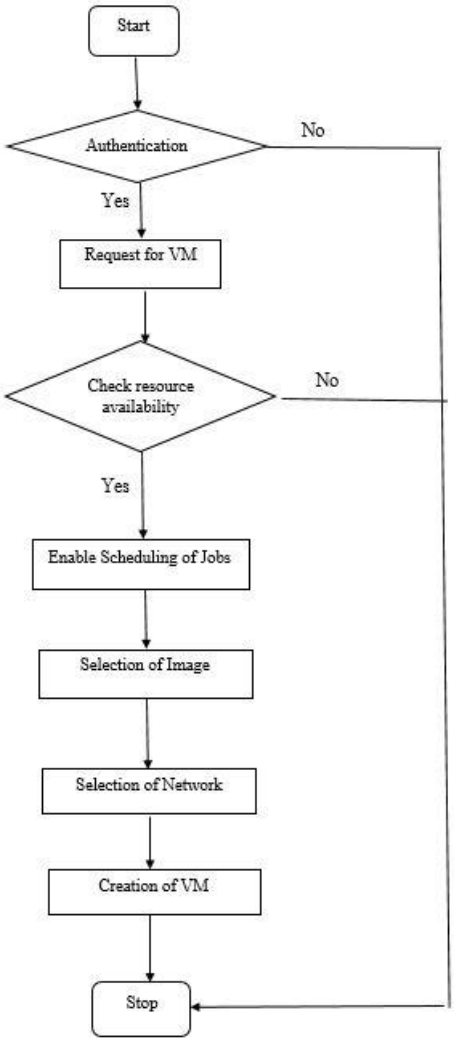

Figure 5: Creating a VM or Instance

This research work can be divided into two main sections as follows:

- Generation of Orphan VMs
- Detection and Elimination of Orphan VMs

3.1.1 Generation of Orphan VMs

Previous studies states that, generation of orphan VMs usually takes place due to the following two reasons i.e.

i. Sudden abortion of a process or activity before completion

This scenario is explained well by the following process flow diagram Figure 6.

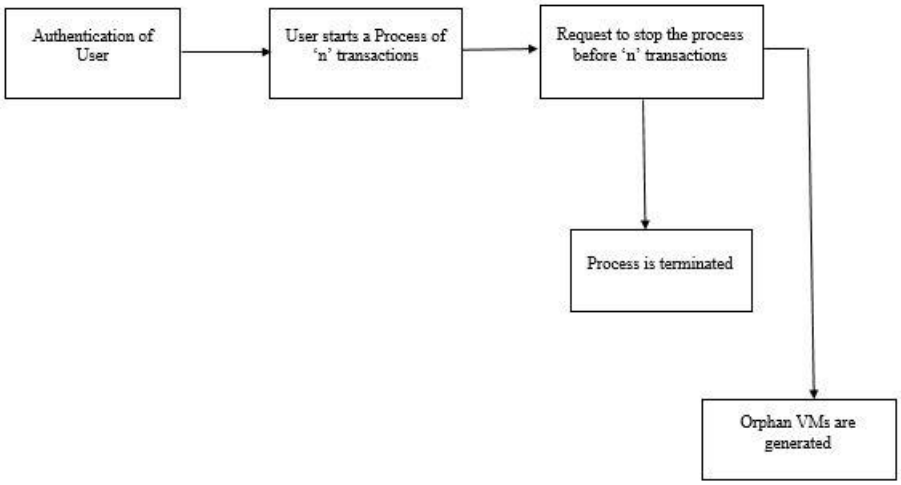

Figure 6: Generation of Orphan VMs-1

It is based on a simple example, it shows that, once the authentication is completed, suppose a user sends a request to start a new process of n number of transactions. Unfortunately, due to certain circumstances, before completion of all the transactions, the user terminates the process. The process will no doubt stop, but there will be a possibility of generation of some orphan VMs due to the sudden abortion.

ii. Unsuccessful deletion of VMs (virtual machines) or instances

In order to understand this possible reason, a flow chart is provided below. The chart is specifically designed by taking OpenStack cloud into consideration. As per the given figure Figure 7, once the authentication is completed using Keystone on OpenStack dashboard, let us suppose that the user requests to delete several VMs (virtual machines) or instances from the Horizon. After this, the scheduling of the jobs with the help of Nova scheduler takes place. Soon after this, two tests are passed, first one is to check the network connectivity and second one is to check load balancing. If proper network connection is enabled, the process is further carried out to check the load balancing. Else, there is a possibility of generation of orphan VMs. In the second test, if load balancing is not enabled, there is again a possibility of generation of orphan VMs. Else, the successful deletion of orphan VMs is carried out.

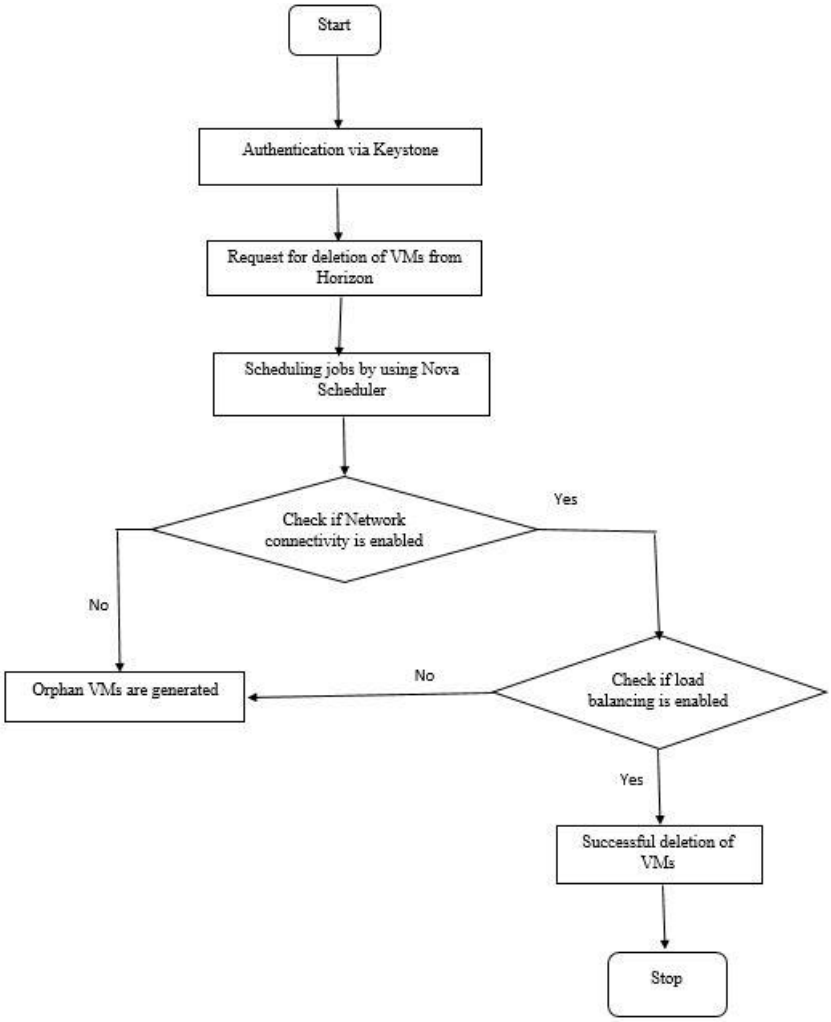

Figure 7: Generation of Orphan VMs-2

3.1.2 Detection and Elimination of Orphan VMs

For detecting that, whether there are orphan VMs present in the database or not, few formulas are used to calculate the availability, total number of resources, etc. The following formula is used to calculate the availability i.e.

Availability of resources = Total Space Used-Guest Usage

We know that, in case of end to end termination of VMs, the VM entries on the dashboard is equal to the VM entries in the database. Hence, total number of resources can be calculated using the formula given below:

Total number of resources = Availability of resources + Occupied Resource

As mentioned above, the objective of this research is to design an automatic process which detects and eliminates the orphan VMs from the database in OpenStack cloud. Hence, an Ansible is created with the help of which orphan VMs will be detected and deleted. The script must be designed on the basis of above given formulas to detect whether there are any orphan VMs created in the database or not. Once, the orphan VMs are detected, they can easily be removed using certain commands in the script. The implementation of this process is described in detail later in the next section of this paper.

3.2 Software Required

This research is carried out by initially configuring OpenStack environment on local system. For this purpose, VMWare Workstation 15 Player is used. A virtual machine named as OpenStack is created on the VMWare Workstation 15 Player and then OpenStack is configured on the VM using DevStack installer. For creating OpenStack development environment, following steps are carried out using the VMWare Workstation 15 Player:

a. Create Ubuntu 16.04 desktop VM.
b. Download DevStack on the VM.
c. Download Rally benchmarking tool.

The Ubuntu 16.04 Linux machine OpenStack has the following specifications:

- Memory: 12GB
- Processors: 1
- Hard Disk: 30GB
- Network Adapter 1: NAT
- Network Adapter 2: Host-only

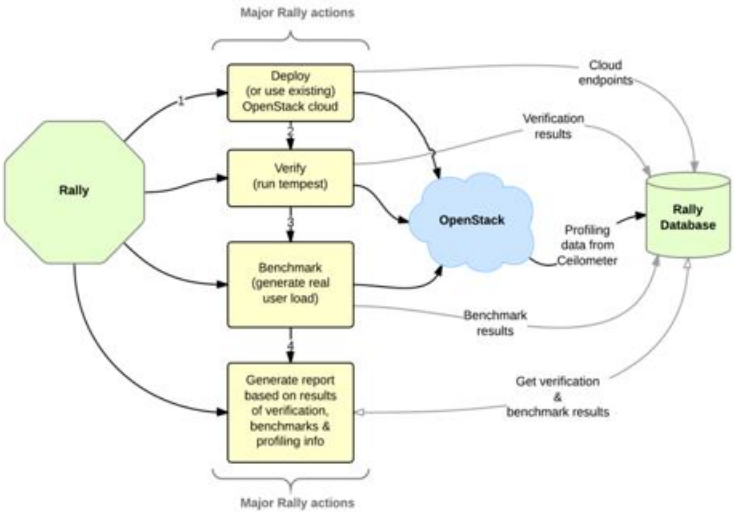

Figure 8: Rally Architecture

3.2.1 DevStack

DevStack is a sequence of scripts with the help of which OpenStack development environment can be created by downloading latest version of required software from git master.

3.2.2 Rally

Rally is a benchmarking tool which is used to test the performance and throughput of the system. It basically evaluates that how the OpenStack cloud is working. Besides this, rally is also used to improve the performance, SLA and stability of OpenStack. Figure 8 describes the architecture of Rally along with the actions performed by it related to OpenStack (n.d.).

4 Implementation

4.1 Continuous Creation and Deletion of VMs

It is already specified in the previous sections of the paper, that the most possible reason for the generation of orphan VMs can be the unsuccessful deletion of VMS. Hence, for this research purpose, continuous creation and deletion of VMs is performed in the OpenStack environment so that, at a certain point, unsuccessful deletion can be encountered and orphan VM can be generated as a result of it. For this purpose, instances are created and deleted continuously by using an Ansible script. This process is done, just to examine whether the solution is helpful in detecting and eliminating the orphan VMs or not. The script is run by using the following command:

./create-vms.sh

Once, the script is made to run, instances are created and deleted after every 5 minutes continuously. Though, it's a very long process, but at a certain point there are some error VMs or orphan VMs generated in the database. The status of all the servers can be checked by using the following command:

- openstack@ubuntu: openstack server list

Following Figure 9 shows the transitions in the states of instances:

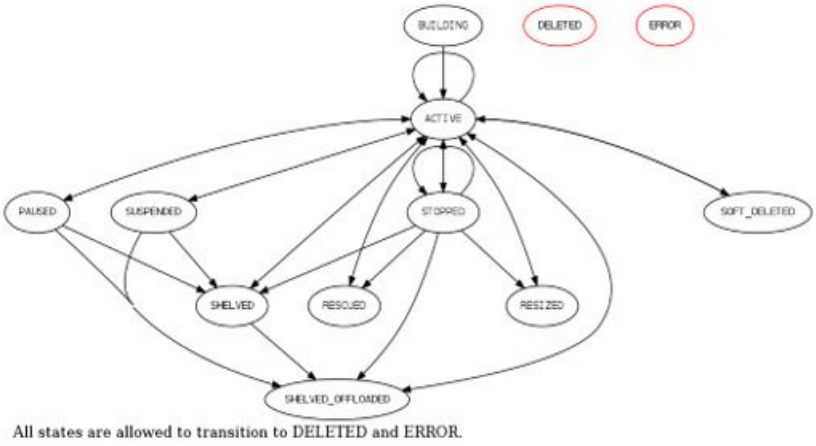

Figure 9: State transitions of instances

4.2 Detection and Elimination of Orphan VMS

The experiment of creating and deleting the VMs was carried out for a whole day, i.e. the Ansible script was made to run several times in a day after a particular period of time. Eventually, some error VMs must have been generated. As specified in the previous sections of the paper, detection process can be carried out manually by recording as well as comparing the VM IDs from both dashboard and database. But it is not feasible when there are a very large number of VMs or resources are present in the cloud. This research eradicates this problem, by providing a solution, i.e. an Ansible script which automatically detects the orphan VMs or error VMs and deletes them from the database.

The script is made to run using the following command:

ansible-playbook cleanup-failed.yml -i hosts –extra-vars="'cat data.json'"

After executing the above command, all the servers present in the database are checked, and any VM with an error state is deleted automatically. Hence, by this process orphan VMs are detected and deleted. This is a continuous process which is made to run after a particular interval. Thus, if at any point of time, orphan VMs or error VMs are generated, they are eliminated immediately, and no resource leakage is caused. Following process flow diagram Figure 10 describes this process more clearly:

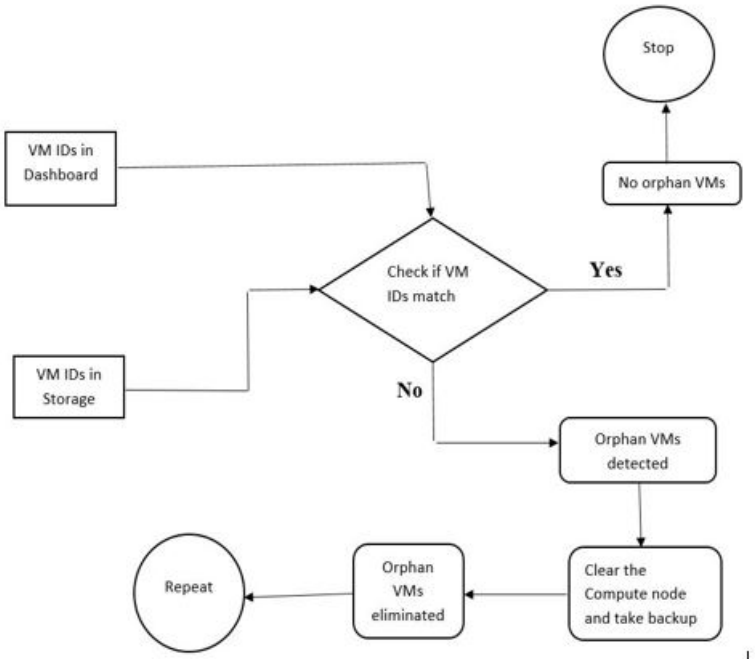

Figure 10: System diagram for Detection and Elimination of Orphan VMS

5 Evaluation

The term evaluation refers to the testing of accuracy and efficiency of the proposed solution. It is the method of checking whether the proposed solution is efficient enough to overcome all the limitations caused due to the generation of orphan VMs. In this research, evaluation of the solution is performed by two ways:

- Comparing the availability of resources before and after implementing the orphan control solution.
- Comparing the performance throughput of the system before and after implementing the orphan control solution.

Given below are some experiments which were performed for recording the availability and system performance before and after implementing the orphan control solution.

5.1 Experiment 1

The first experiment is carried out in order to test the availability of resources for a whole day after particular intervals of time. This experiment was performed before the implementation of orphan control method. This means the availability of resources is evaluated here in the presence of orphan VMs database. The table given below depicts the output of this experiment:

Time	Instances	Volumes	VCPUs	Volume Storage
9:00	10	10	10	10
11:00	8	9	9	8
13:00	6	7	8	8
15:00	7	8	9	7
17:00	5	7	8	6
19:00	8	8	9	9

Following table depicts the availability of resources at an interval of two hours

Time	Availability
9:00	40
11:00	34
13:00	29
15:00	31
17:00	26
19:00	34

The graphs given below Figure 11 and Figure 12 are plotted against availability and time before implementing the orphan control method:

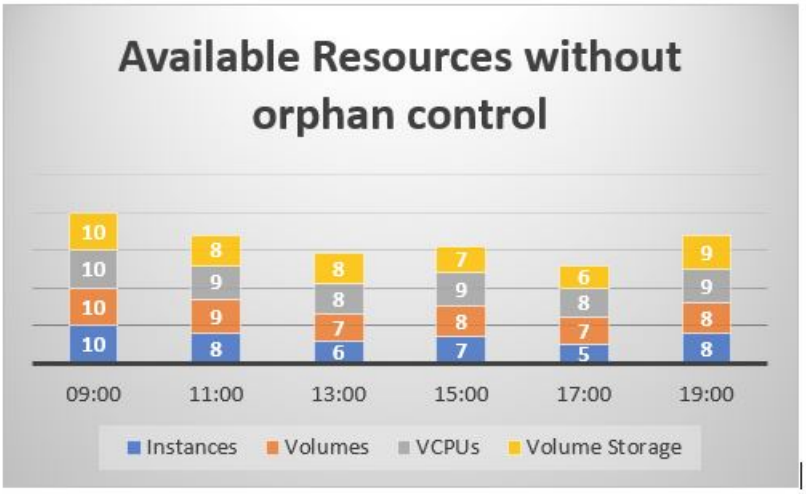

Figure 11: Available Resources before implementing orphan control solution.

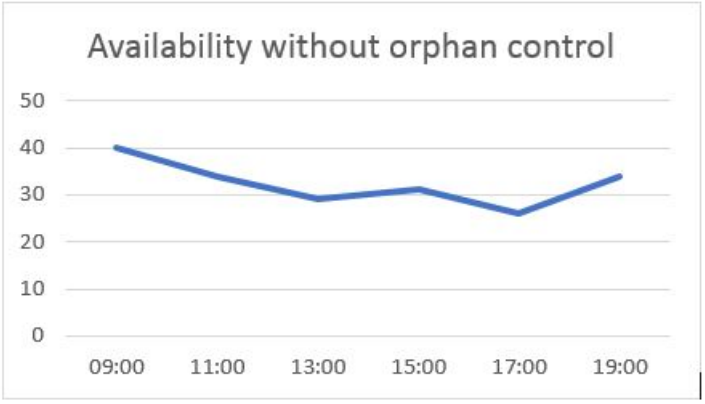

Figure 12: Availability before implementing orphan control

5.2 Experiment 2

This experiment is carried out in order to test the availability of resources for a whole day after particular intervals of time. This experiment was performed after the implementation of orphan control method. This means the availability of resources is evaluated here with no orphan VMs present in the database. The table given below depicts the output of this experiment:

Time	Instances	Volumes	VCPUs	Volume Storage
9:00	10	10	10	10
11:00	8	9	9	9
13:00	7	8	9	7
15:00	9	9	8	9
17:00	8	7	9	8
19:00	10	10	10	10

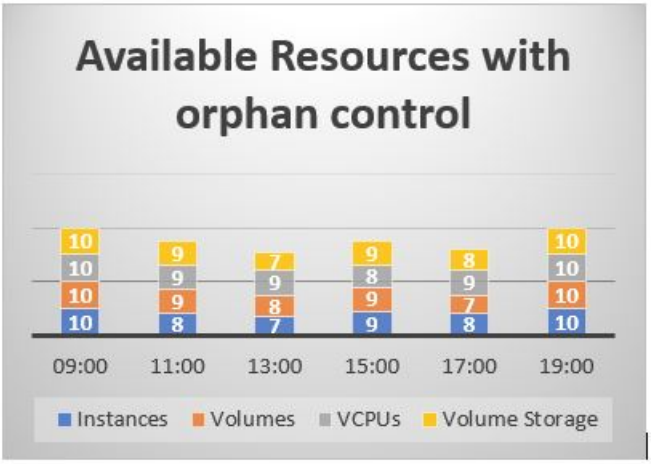

Figure 13: Available Resources after implementing orphan control solution.

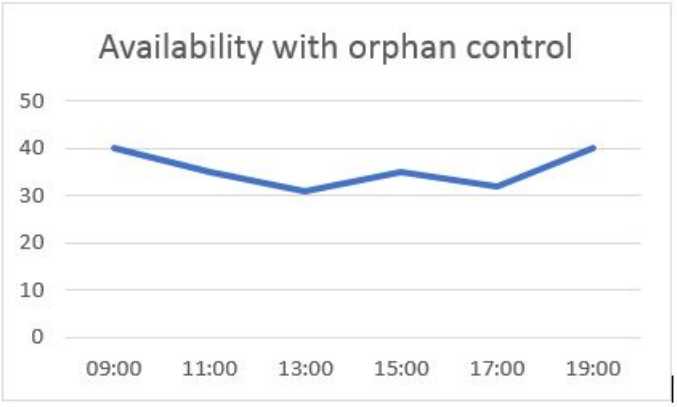

Figure 14: Availability after implementing orphan control solution.

The above given graph Figure 13 depicts the available resources before implementing the solution at an interval of two hours. Following table indicates the recorded availability after every two hours on the day of experiment. The graph provided above Figure 14 resembles the following table and is plotted against availability and time.

Time	Availability
9:00	40
11:00	35
13:00	31
15:00	35
17:00	32
19:00	40

5.3 Comparison of availability of resources

This section depicts the qualitative as well as quantitative comparison of the availability of resources in openstack cloud before and after implementing the orphan control.

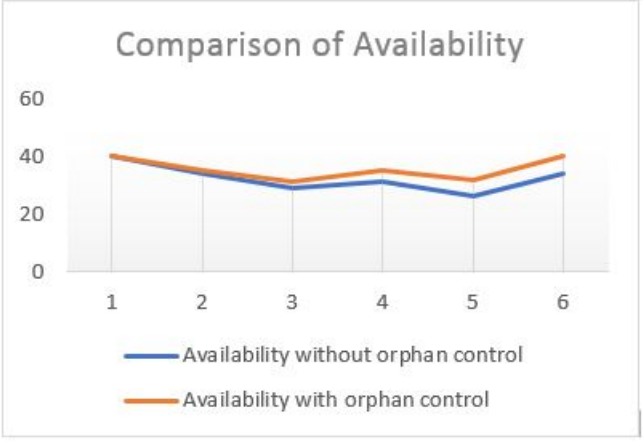

Figure 15: Comparison of Availability

The following table lists the availability of resources in both the cases i.e. before implementing the orphan control solution and after implementing the orphan control solution.The graph Figure 15 provided above depicts the comparison of availabilities in both the cases.

Availability without Orphan Control	Availability with Orphan Control
40	40
34	35
29	31
31	35
26	32
34	40

The above graph Figure 15 evaluates that, the availability with orphan control solution is higher than that of without orphan control solution. Hence, the orphan control solution helps in improving the availability of resources.

5.4 Experiment 3

This experiment is carried out in order to test the system performance. For this purpose, a benchmarking software tool known as rally is used. This experiment was performed before the implementation of orphan control method. This means the system performance is evaluated here in the presence of orphan VMs database. The table given in the following snapshot depicts the output of this experiment:

Response Times (sec)

Action	Min (sec)	Median (sec)	90%ile (sec)	95%ile (sec)	Max (sec)	Avg (sec)	Success	Count
nova.boot_server	129.799	181.397	185.219	186.112	187.004	166.955	100.0%	10
nova.delete_server	9.675	13.223	34.698	34.924	35.15	19.051	100.0%	10
total	152.978	193.345	197.718	197.853	197.987	186.006	100.0%	10
-> duration	151.978	192.345	196.718	196.853	196.987	185.006	100.0%	10
-> idle_duration	1.0	1.0	1.0	1.0	1.0	1.0	100.0%	10

Figure 16: Output of rally without orphan control

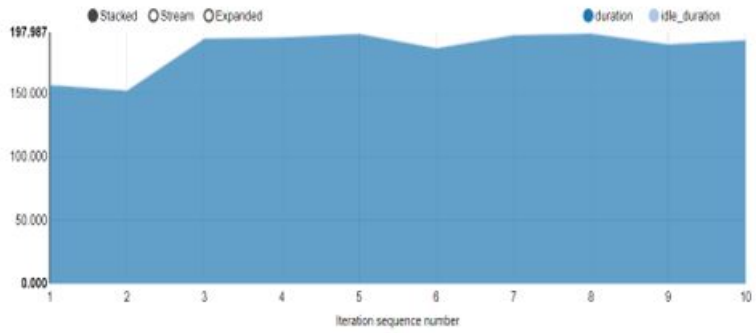

Figure 17: Load Duration

- Load Duration: 197.234
- Full Duration: 215.275

Rally also generates related graphs Figure 17 and Figure 18 which are provided here. These graphs depict the performance, load duration and distribution without orphan control solution:

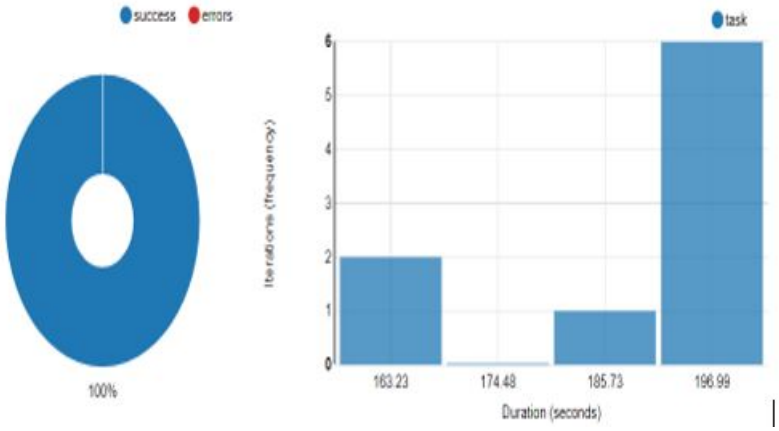

Figure 18: Distribution

Response Times (sec)								
Action	Min (sec)	Median (sec)	90%ile (sec)	95%ile (sec)	Max (sec)	Avg (sec)	Success	Count
nova.boot_server	56.46	75.854	126.374	131.857	137.339	85.906	100.0%	10
nova.delete_server	3.287	37.875	47.203	47.46	47.716	32.202	100.0%	10
total	102.221	114.497	137.814	139.22	140.626	118.109	100.0%	10
-> duration	101.221	113.497	136.814	138.22	139.626	117.109	100.0%	10
-> idle_duration	1.0	1.0	1.0	1.0	1.0	1.0	100.0%	10

Figure 19: Output of Rally with Orphan Control

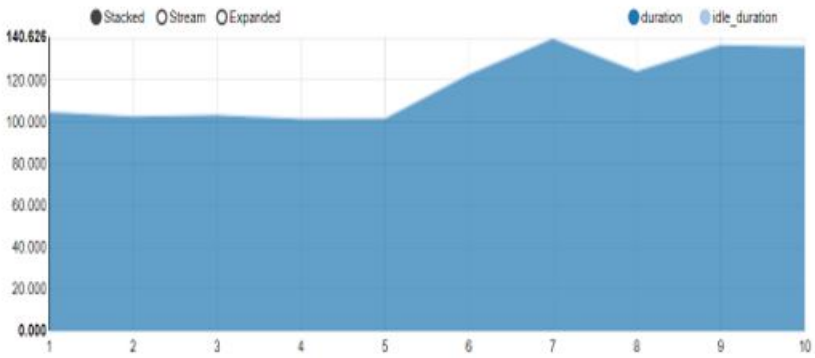

Figure 20: Load Duration

5.5 Experiment 4

This experiment is carried out in order to test the system performance using a benchmarking software tool known as rally. This experiment was performed after the implementation of orphan control method. This means the system performance is evaluated here in the presence of orphan VMs database. The table given in the above snapshot Figure 19 depicts the output of this experiment:

- Load Duration: 139.822
- Full Duration: 156.38

Rally also generates related graphs Figure 20 and Figure 21 which are provided here. These graphs depict the performance, load duration and distribution with orphan control solution:

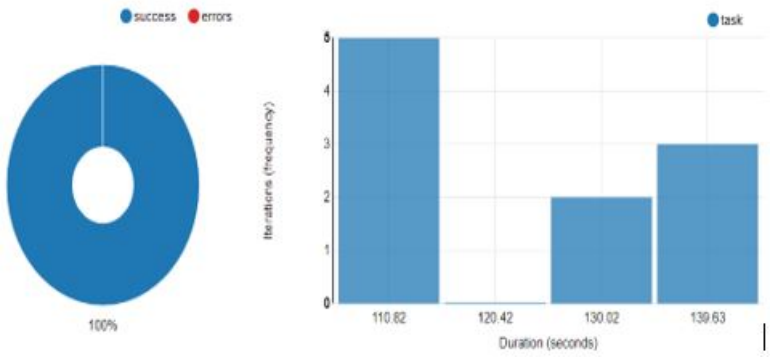

Figure 21: Distribution

5.6 Discussion

It is already evaluated in section 5.3, that the orphan control solution improves the availability of the resources in OpenStack cloud. Now, coming to system performance, rally is used to check the performance of the system in both the cases. By examining the output tables and graphs generated in experiments 3 and 4, it is evaluated by rally that the response time is less after implementing the orphan control solution. This means the performance of the system is better in case of with orphan control rather than without orphan control.

Limitations of the orphan control solution:

- In this research, OpenStack environment is configured using devstack. Here, all the three nodes of OpenStack cloud are resembled as a single node. So, its unidentified that the orphan control solution is effective enough in the companies using OpenStack and not devstack.
- As here devstack (all in one) is used, so there is no needs of synchronization of nodes, but if this solution is to be implemented in proper OpenStack environment, node synchronization need to be enabled.
- The hypervisor QEMU is used in this research for configuring the OpenStack environment, hence it is unidentified whether there is any change in the performance of orphan control solution with the change of hypervisor.

6 Conclusion and Future Work

As specified in the abstract, the objective of this paper is to eradicate the limitations caused due to the generation of orphan VMs in OpenStack cloud platform. This research paper describes all the steps involved in implementing the solution to the problem of orphan VMs. Ansible is used to write the coding scripts, with the help of which the solution to detect and eliminate orphan VMs is successfully implemented. The results generated in the output are also provided and described in the paper. Along with this, any further generation of orphan VMs can also be terminated automatically, because it is a continuous process. This periodic nature of the solution makes it more effective and unique. Moreover, this solution provides benefits such as increased availability of resources, improved system performance and helps in decreasing cost expenses.

For future work of the research, the solution should be sent to the OpenStack community to add this orphan control solution code to their repository. So, that in the future, whenever, devstack is downloaded from the git-hub repository, the orphan control code is already present. In this way, the problem of orphan VMs can be vanished completely.

7 Appendix- Configuration Manual

The research problem taken into consideration is the limitations caused due to the generation of orphan VMs in the database of a cloud. The major disadvantages that orphan VMs create are:

- Decreases the availability of resources.
- Degrades the performance of the system.
- It may also result in system failure.

In order to overcome all these limitations, an orphan control solution is proposed in this research which detects if there is any orphan VM present in the database of the cloud and automatically eliminates the orphan VMs if detected. OpenStack cloud platform is taken into consideration for carrying out the research.

The configuration part of the research basically consists of the following sections:

7.1 Software Required

As specified above, this research is performed in the OpenStack cloud platform. Hence, the first step is to configure the OpenStack cloud environment on the local system. For this purpose, VMWare Workstation 15 Player is used. A virtual machine named as OpenStack is created on the VMWare Workstation 15 Player and then OpenStack is configured on the VM using DevStack installer. For creating OpenStack development environment, following steps are carried out:

7.2 Installation of DevStack and Rally

7.2.1 DevStack

DevStack is a sequence of scripts with the help of which OpenStack development environment can be created by downloading latest version of required software from git master.

7.2.2 Rally

Rally is a benchmarking tool which is used to test the performance and throughput of the system. It basically evaluates that how the OpenStack cloud is working. Besides this, rally is also used to improve the performance, SLA and stability of OpenStack.

7.2.3 Installation Steps

After creating the Linux machine OpenStack successfully, DevStack and Rally are downloaded by carrying out the following steps:

- sudo useradd -s /bin/bash -d /opt/stack -m stack
- echo ”stack ALL=(ALL) NOPASSWD: ALL” — sudo tee /etc/sudoers.d/stack

- reboot
- sudo su-stack
- sudo apt-get install git
- git clone https://git.openstack.org/openstack-dev/devstack
- git clone https://github.com/openstack/rally
- cd devstack
- sudo apt-get install ansible
- sudo apt-get install python-pip
- pip install shade
- cp samples/local.conf local.conf
- enable-plugin rally https://github.com/openstack/rally master
- Thus, OpenStack environment is configured successfully. By running the following command, the ip address at which OpenStack is configured is provided:

  ./stack.sh

- OpenStack dashboard can be accessed in the web browser (Mozilla Firefox) of the virtual machine. Two users of OpenStack are created i.e. admin and demo. Ping the following address using the credentials in the browser:

  192.168.206.132/dashboard

7.3 Common Errors

Following are the common errors one can face in installing devstack:

- Permission denied error:
  It can be solved by using the following command:
  sudo chown -R stack:stack /opt/stack
- Environment Error
  To solve this error, following command is used:
  virtual env../requirements/.venv/

7.4 For evaluating Availability of Resources in OpenStack

After installing devstack, following steps must be carried out:

- Ping the following address using the credentials in the browser.

192.168.206.132/dashboard

- Download the demo.rc.(V3) file from the dashboard.
- Open a new terminal.
- Change directory to downloads

  cd downloads

- cp demo-openrc.sh/opt/stack
- sudo su -stack
- Edit the file

  nano /.rally/openrc

- source demo-openrc.sh

```
export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_NAME=demo
export OS_TENANT_NAME=demo
export IMAGE=cirros-0.3.5-x86_64-disk
export STATE=present


#!/bin/sh
for i in 1 .. 12
do
        echo "Executing for $i time(s).. "
        echo "Creating the VM(s) ... "
        cat<<EOF > data.json
{
    "osAuthURL"     : "$OS_AUTH_URL",
    "username"      : "$OS_USERNAME",
    "password"      : "$OS_PASSWORD",
    "projectName"   : "$OS_PROJECT_NAME",
    "userDomainName"        : "Default",
    "imageName"     : "$IMAGE",
    "keyName"       : "akanksha",
    "flavor"        : "1",
    "state"         : "$STATE",
    "vms": [
              {
                 "hostName" : "test1"
              },
              {
                 "hostName" : "test2"
              },
              {
                        "hostName" : "test3"
               },
              ]
}
EOF
        ansible-playbook openstack.yml -i hosts --extra-
vars="`cat data.json`"

        echo "Deleting the VM(s) ... "
        STATE=absent
        cat<<EOF > data.json
{
    "osAuthURL"     : "$OS_AUTH_URL",
    "username"      : "$OS_USERNAME",
    "password"      : "$OS_PASSWORD",
    "projectName"   : "$OS_PROJECT_NAME",
    "userDomainName"        : "Default",
    "imageName"     : "$IMAGE",
        "keyName"       : "akanksha",
        "flavor"        : "1",
        "state"         : "$STATE",
        "vms": [
        {
          "hostName" : "test1"
         },
     {
                "hostName" : "test2"
     },
     {
                "hostName" : "test3"
     },
   ]
}
EOF
        ansible-playbook openstack.yml -i hosts --extra-
vars="`cat data.json`"

        sleep 300
done
```

Above given ansible script is used for continuous creation and deletion of instances. Following command is used to run the above code:

./create-vms.sh

```
---
- name: Deploy guest from template
  hosts: local
  gather_facts: no
  tasks:
    - os_server_facts:
        server: test*

    - name: Create/Delete list of instance(s) and attaches to a
network and passes metadata to the instance
      os_server:
        state: "absent"
        auth:
          auth_url: "{{ osAuthURL }}"
          username: "{{ username }}"
          password: "{{ password }}"
          project_name: "{{ projectName }}"
          user_domain_name: "{{ userDomainName }}"
        name: "{{ item.name }}"
        timeout: 200
      when: "{{ item.status = 'ERROR' }}"
      with_items:
        - "{{ openstack_servers }}"
```

The above given ansible script is used to detect and eliminate orphan VMs from the database. Following command is used to run the above code:

ansible-playbook cleanup-failed.yml -i hosts –extra-vars="'cat data.json'"

7.5 For running Rally

For running the software benchmarking tool rally, following steps are carried out:

- cd rally/samples/tasks/scenarios
- Edit the boot-and-delete.json file

  vi boot-and-delete.json

  Copy the code into this file.
- cd
- source demo-openrc.sh
- rally deployment check
- rally task start rally/samples/tasks/scenarios/boot-and-delete.json
- Rally automatically generates the tables and graphs which depicts the load duration and distribution.

```
    {
      "title": "NovaServers.boot_and_delete_server",
      "description": "Boot and delete a server.",
      "scenario": {
        "NovaServers.boot_and_delete_server": {
          "force_delete": false,
          "flavor": {
            "name": "m1.tiny"
          },
          "image": {
            "name": "^cirros.*-disk$"
          }
        }
      },
      "contexts": {
        "users": {
          "users_per_tenant": 2,
          "tenants": 3
        }
      },
      "runner": {
        "constant": {
          "concurrency": 10,
          "times": 10
        }
      },
      "hooks": [],
      "sla": {
        "failure_rate": {
          "max": 0
        }
      }
    }
  ]
}
```

The above given code is written in the boot-and-delete.json file. And rally is initiated by using the following command:

rally task start rally/samples/tasks/scenarios/boot-and-delete.json

References

(n.d.).

Barbonis, P. A. (2006). Towards some theory of technology orphaning: A retrospective exploratory study, *IEEE* (5): 1–3.

Barkat, A., dos Santos, A. D. and di Milano, T. T. N. H. (2014). Openstack and cloudstack: Open source solutions for building public and private clouds, *IEEE* **8**(8): 1–5.

Dabrowski, C. and Mills, K. (2011). Vm leakage and orphan control in open-source clouds, *IEEE* **6**(6): 1–3.

Gong, W., Chen, Z. and Yan, J. (2014). An optimal vm resource allocation for nearclient-datacenter for multimedia cloud, *IEEE* (6): 1–2.

guang Ao, Z., hai Jiao, M., ning Gao, K. and wei Wang, X. (2016). Research on cloud resource optimization model based on users' satisfaction, *IEEE* (4): 1–2.

Gugnani, S., Lu, X. and Panda, D. K. D. (2017). Swift-x: Accelerating openstack swift with rdma for building an efficient hpc cloud, *IEEE* (10): 1–4.

HERLIHY, M. P., MEMBER, I. and McKENDRY, M. S. (1989). Timestamp-based orphan elimination, *IEEE* **15**(8): 1–2.

Jahanshahi, M., Mostafavi, K., Kordafshari, M. S., Gholipour, M. and Haghighat, A. (n.d.). Two new approaches for orphan detection, *IEEE* **1**.

Jain, P., Datt, A., Goel, A. and Gupta, S. C. (2016). Cloud service orchestration based architecture of openstack nova and swift, *IEEE* (7): 1–6.

Konoor, D. K. (n.d.). Auditing in cloud computing solutions with openstack, *IEEE* **29**.

Nou, R., Miranda, A., Siquier, M. and Cortes, T. (2017). Improving openstack swift interaction with the i/o stack to enable software defined storage, *IEEE* (8): 1–2.

Paper, W. (2014). Cloud computing innovation in india: A framework and roadmap, *IEEE* **208**(208): 30–90.

Parikh, S. M., Patel, D. N. M. and Prajapati, D. H. B. (2017). Resource management in cloud computing: Classification and taxonomy, *airxiv* (10): 1–4.

Ruan, M., Titcheu, T., Zhai, E., Li, Z., Liu, Y., E, J., Cui, Y. and Xu, H. (n.d.). On the synchronization bottleneck of openstack swift-like cloud storage systems, *IEEE* **29**.

Teixeira, J. (2014). Developing a cloud computing platform for big data: The openstack nova case, *IEEE* (3): 1–2.

Theng, D. and Hande, K. N. (2012). Vm management for cross-cloud computing environment, *IEEE* (5): 1–2.

Xu, Q. and Yuan, J. (2014). A study on service performance evaluation of openstack, *IEEE* (4): 1–2.

dm Mann, Z. (2017). Resource optimization across the cloud stack, *IEEE* **29**(17): 1–2.