Comparison and Evaluation of Open-source Cloud Management Software

SRIVATSAN JAGANNATHAN

Masters' Degree Project
Stockholm, Sweden
XR-EE-LCN 2012:008
Comparison and Evaluation of Open-source Cloud Management software

M.Sc Thesis

Information and Communication Systems Security

Srivatsan Jagannathan

Examiner/Advisor: Prof. Rolf Stadler

Project Supervisor: Rerngvit Yanggtatoke

Laboratory for Communication Networks

School of Electrical Engineering

KTH Royal Institute of Technology, Stockholm, Sweden

June 2012
ACKNOWLEDGEMENTS

I would like to thank Professor Rolf Stadler for allowing me to do this thesis at the Laboratory of Communication Networks (LCN) and my supervisor Rerngvit Yanggratoke for his support during this thesis work. I would also like to thank my friends and family for their continuous support and encouragement.
ABSTRACT

The number of cloud management software related to a private infrastructure-as-a-service cloud is increasing day-by-day. The features of the cloud management software vary significantly and this creates a difficulty for the cloud consumers to choose the software based on their business requirements. An example of the problem is choosing software with a power management feature. The power management feature is used to increase the efficiency of energy consumption by consolidating virtual machines together and turning off unused physical servers, which is not provided by many cloud management software.

OpenNebula is one of the most widely used open-source cloud management software among research institutions and enterprises. However, the performance characteristic of OpenNebula is not well studied in the existing literature. An example of the problem is choosing a hardware configuration to run OpenNebula for the research institutions and enterprises.

The first objective of this thesis is to develop a framework for comparing features of various cloud management software. For developing this framework, existing works are reviewed. The cloud management software is installed on the KTH LCN testbed for hands-on experience. Both the open-source and the commercial software are analyzed for developing the framework. The major contribution related to the framework is identifying features provided for the commercial software that are not available for the open-source software. The features are: (1) co-location of VMs is running a group of VMs on the same physical machine (for example, if the web server VM has to access the application server VM for getting the web pages, they can be placed on the same physical machine); (2) anti-co-location of VMs is not allowing a pair of VMs to run on a single physical machine (for example, the primary and back-up web server VMs should always run on different physical machines); (3) the resources of the physical machines can be combined (e.g., number of CPU cores, physical memory) as a resource pool and compartmentalized into an organizational structure (e.g., HR, development, testing, etc).

The second objective of this thesis is to evaluate the performance of the OpenNebula cloud management software. For the performance evaluation, existing works are reviewed to identify the metrics, and the OpenNebula cloud management software is installed on the KTH LCN testbed. The performance of the OpenNebula software was evaluated for different virtual machine operations, virtual machine types, number of virtual machines and change in load of the system. The major lessons learned related to the performance evaluation are: (1) the duration for the live migration does not change with the load; (2) the duration for the live migration increases linearly as the memory assigned to the VM increases; (3) the duration of the add and delete operations increases linearly as the number of VMs increases.

Keywords: cloud management software, framework, metrics, performance evaluation.
# Table of Contents

1 Introduction .................................................................................................................. 1  
  1.1 Background and Motivation ................................................................................. 1  
  1.2 Problem Definition ............................................................................................... 2  
  1.3 The Approach ....................................................................................................... 3  
  1.4 Contribution of the Thesis ................................................................................... 3  
  1.5 Outline of Thesis ................................................................................................. 4  

2 Related Works ............................................................................................................. 5  

3 Background .................................................................................................................. 8  
  3.1 Cloud Computing .................................................................................................. 8  
    3.1.1 Definition of Cloud Computing ....................................................................... 8  
    3.1.2 Characteristics of Cloud Computing ............................................................. 8  
    3.1.3 Service Models .............................................................................................. 8  
    3.1.4 Deployment Models ...................................................................................... 9  
  3.2 Physical Machine .................................................................................................. 9  
  3.3 Virtual Machine ................................................................................................... 9  
  3.4 Network ................................................................................................................. 9  
  3.5 Virtual Machine Image ......................................................................................... 10  
  3.6 Hypervisor ........................................................................................................... 10  
  3.7 Cloud Management Software .............................................................................. 11  
  3.8 Zabbix software ................................................................................................... 12  
  3.9 SOAP and REST ................................................................................................. 13  

4 Framework for Comparing Features of Cloud Management Software ..................... 14  
  4.1 Basic Information ................................................................................................. 14  
  4.2 Software Architecture ......................................................................................... 14  
    4.2.1 Cloud Architecture Presentation ................................................................... 14  
    4.2.2 Eucalyptus Architecture ............................................................................. 15  
    4.2.3 Nimbus Architecture ................................................................................... 16  
    4.3.4 CloudForms Architecture ............................................................................ 17  
    4.2.5 OpenStack Cactus Architecture ................................................................. 18  
  4.2.6 Abiquo Architecture ....................................................................................... 19
5.8.2 Comparing Free Memory for Adding Virtual Machine in the No-load and the Load Setups .................................................................................................................. 44
5.8.3 Comparing Free Memory for Adding Different Virtual Machine Types in the No-load Setup .................................................................................................................. 46
5.8.4 Comparing Free Memory for Adding Different Numbers of Virtual Machines in the No-load Setup .................................................................................................................. 47
5.8.5 Analyzing Redundant Memory for 2 Large-Instance Virtual Machines and 8 Small-Instance Virtual Machines in the No-load setup .................................................................................................................. 49
5.8.6 Evaluating Free Memory for Deleting Virtual Machines in the No-load Setup ....... 50
5.8.7 Comparing Free Memory for Deleting Virtual Machine in the No-load and the Load Setups .................................................................................................................. 51
5.8.8 Comparing Free Memory for Deleting Different Virtual Machine Types in the No-load Setup .................................................................................................................. 53
5.8.9 Comparing Free Memory for Deleting Different Numbers of Virtual Machines in the No-load Setup .................................................................................................................. 54
5.8.10 Analyzing Redundant Memory for 2 Large-Instance Virtual Machines and 8 Small-Instance Virtual Machines in the No-load setup .................................................................................................................. 56
5.8.11 Evaluating Network Transfer for the Live Migration of Virtual Machines in No-load Setup .................................................................................................................. 57
5.8.12 Comparing Network Transfer for the Live Migration of Virtual Machines in the No-load and Load Setups .................................................................................................................. 59
5.8.13 Comparing Network Transfer for the Live Migration of Different Virtual Machine Types in the No-load Setup .................................................................................................................. 60
5.8.14 Evaluating Network Transfer for the Cold Migration of Virtual Machine in the No-load Setup .................................................................................................................. 62
5.8.15 Comparing Cold Migration and Live Migration of Virtual Machine in the No-load Setup .................................................................................................................. 64
5.8.16 Comparing Network Transfer for the Cold Migration of Virtual Machine in the No-load Setup and Load Setup .................................................................................................................. 65
5.8.17 Comparing Network Transfer for the Cold Migration of Different Virtual Machine Types in the No-load Setup .................................................................................................................. 67
5.9 Overall Conclusion .................................................................................................................. 68
5.9.1 Major Contribution .................................................................................................................. 68
6 Conclusions and Future Work ........................................................................................................ 73
6.1 Conclusion .......................................................................................................................... 73
6.2 Experiences ....................................................................................................................... 73
6.3 Future Work ..................................................................................................................... 74
References ............................................................................................................................. 75
Appendix A ............................................................................................................................. 84
# Table of Figures

- Figure 1: Full virtualization (a) and hardware-layer virtualization (b) models [7] ........................................ 10  
- Figure 2: Interaction between the cloud management software and other components [7] .................... 11  
- Figure 3: Zabbix setup ................................................................................................................................. 12  
- Figure 4: The architecture of the Eucalyptus cloud management software [33] ........................................ 16  
- Figure 5: The architecture of the Nimbus cloud management software [53] ............................................. 17  
- Figure 6: The software components of the CloudForms [110] .................................................................... 17  
- Figure 7: The Architecture of the OpenStack cloud management software [65] ........................................ 19  
- Figure 8: The Architecture of the Abiquo cloud management software [63] ............................................. 19  
- Figure 9: The architecture of the openQRM cloud management software [111] ....................................... 20  
- Figure 10: The architecture of the VMware infrastructure [86] ................................................................. 21  
- Figure 11: The architecture of the OpenNebula cloud management software [107] ............................... 22  
- Figure 12: Testbed for no-load setup ........................................................................................................... 38  
- Figure 13: Testbed for load setup ................................................................................................................ 39  
- Figure 14: Faban Setup ................................................................................................................................. 41  
- Figure 15: Free memory for adding 2 SIVMs in the no-load setup ............................................................ 43  
- Figure 16: Free memory for adding 2 SIVMs in the no-load and load setup .............................................. 45  
- Figure 17: Free memory for adding 2 SIVMs and 2 LIVMs in the no-load setup ...................................... 46  
- Figure 18: Free memory for adding 2 SIVMs and 8 SIVMs in the no-load setup ...................................... 48  
- Figure 19: Free Memory for Adding 2 LIVMs and 8 SIVMs in the no-load setup ..................................... 49  
- Figure 20: Free memory for deleting 2 SIVMs in the no-load setup ......................................................... 51  
- Figure 21: Free memory for deleting 2 SIVMs in the no-load and load setup ........................................ 52  
- Figure 22: Free memory for deleting 2 SIVMs and 2 LIVMs in the no-load setup .................................. 54  
- Figure 23: Free memory for deleting 2 SIVMs and 8 SIVMs in the no-load setup .................................. 55  
- Figure 24: Free memory for deleting 2 LIVMs in no-load setup and 8 SIVMs in the no-load setup .......... 57  
- Figure 25: Network transfer per second for the live migration of 8 SIVMs in the no-load setup from Server 2 to Server 1 ................................................................. 58  
- Figure 26: Network transfer per second for the live migration of 8 LIVMs from Server 2 to Server 1 in the no-load (a) and load setup (b) .......................................................... 59  
- Figure 27: Network transfer per second for the live migration of 8 SIVMs (a) and 8 LIVMs (b) from Server 2 to Server 1 in the no-load setup ......................................................... 61  
- Figure 28: Network transfer per second for the cold migration of 8 LIVMs in the no-load setup ............ 62  
- Figure 29: Network transfer per second for the cold migration (a) and the live migration (b) of 8 LIVMs in the no-load setup ............................................................................. 64  
- Figure 30: Network transfer per second for the cold migration of 8 SIVMs in the no-load (a) and load setup (b) from Server 1 to Server 2 ................................................................. 66  
- Figure 31: Network transfer per second for the cold migration of 8 SIVMs (a) and 8 LIVMs (b) from Server 1 to Server 2 in the no-load setup ......................................................... 67
List of Tables
Table 1: Basic information about different cloud management software ........................................ 14
Table 2: Interface representation of the cloud management software architecture .......................... 15
Table 3: Physical host management for different cloud management software ............................ 23
Table 4: Virtual machine management for different cloud management software .......................... 24
Table 5: Virtual machine image management for different cloud management software .............. 25
Table 6: Virtual network management for different cloud management software ......................... 26
Table 7: Cloud interface provided by different cloud management software ............................... 28
Table 8: User management for different cloud management software ........................................ 29
Table 9: Hypervisor supported by different cloud management software ..................................... 30
Table 10: Scheduling provided by different cloud management software ...................................... 31
Table 11: Quota provided by different cloud management software ............................................. 34
Table 12: Hybrid Support provided by different cloud management software .............................. 34
Table 13: Contextualization supported by different cloud management software ......................... 35
Table 14: List of software required to run OpenNebula ................................................................. 40
Table 15: Metrics used for performance evaluation ................................................................. 41
Table 16: Virtual machine management commands executed on OpenNebula ............................ 42
Table 17: Summary of observations for adding virtual machine (VM) ........................................ 69
Table 18: Summary of observations for deleting virtual machine (VM) ...................................... 70
Table 19: Summary of observations for live migration of virtual machine (VM) .......................... 71
Table 20: Summary of observations for cold migration of virtual machine (VM) ......................... 72
1 Introduction

1.1 Background and Motivation

According to NIST, “Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [21].

Cloud computing is a method for delivering different kinds of services over the Internet. It can be a software application (e.g., Gmail) delivered over the Internet. It can be a platform (e.g., Google App Engine) on top of which users can write an application. It can be an operating system (e.g., Amazon EC2 provides OS) where users can create their own platform and software. All these services can be accessed through an Internet connection [1].

Cloud computing solves the demand of computing resources (e.g., CPU, RAM, etc) for processing jobs. In recent years, the demand for processing jobs has increased to hundreds or thousands of CPU cores.

Infrastructure-as-a-Service (IaaS) cloud is one of the models of cloud computing used to deliver computing resources, and this is accomplished by provisioning virtual resources to customers. For example, if a person wants to run an online business, there is no need to set up his/her own IT infrastructure. Instead he/she can rent a virtual machine from Amazon and provide the service [8].

In addition to delivering the virtual resources, IaaS brings another advantage of “pay-per-use”. The pay-per-use model can be easily understood through comparison with an electricity payment scheme, paying only for the amount of electricity consumed. Similarly, IaaS allows a customer to pay only for the used computing resource. This helps the business as it no longer requires investing capital in IT infrastructure and allows an enterprise to focus more on the business growth instead of thinking about over-provisioning and under-provisioning of the computing resource [1].

The actors involved in cloud computing related to IaaS are called cloud stakeholders. The cloud stakeholders are cloud providers and cloud users. The cloud providers are responsible for setting up the IT infrastructure and the cloud users use the infrastructure provided to complete their tasks.

The private-IaaS-cloud model is selected where the cloud infrastructure is provided by a single organization for its different business units. The cloud management software is a piece of software that converts the existing physical IT infrastructure into a cloud infrastructure that supports the private-IaaS-cloud. The number of commercial cloud management software
increases as the popularity of private-IaaS-cloud increases, such as VMware vCloud, CloudForms, etc [9].

Numerous contributions have been made by the open-source community related to private-IaaS-cloud. Some of the open-source cloud management software are Eucalyptus, Nimbus, XCP and OpenNebula [4, 8]. [5, 6] list some open-source private IaaS cloud management software.

The number of cloud management software is increasing day-by-day. One of the major problems for cloud consumers is choosing the cloud management software, as features of the cloud management software vary significantly. This makes it difficult for the cloud consumers to choose the software based on their business requirements. For example, if a cloud customer thinks that power management is important for his/her business, the cloud consumer have to read the information provided in the software website to identify whether the power management is provided with the software or not.

To solve the problem of choosing the cloud management software, a framework needs to be developed. The framework consolidates the features provided by the different cloud management software. Both the open-source and the commercial cloud management software related to the private-IaaS-cloud are included for comparison. This framework provides three advantages. Firstly, it allows cloud consumers to choose software that suits their business requirements. Secondly, cloud management software vendors (the companies that sell their software, e.g., Eucalyptus, OpenNebula, etc.), can include additional features to their software. Finally, new cloud management software developers (one who starts creating a new cloud management software) can start building their software by including the features listed in the framework.

The OpenNebula cloud management software is one of the popular private cloud management software. In particular, the number of cloud consumers downloading OpenNebula software is several thousand per month [3]. Some of the existing works compare the features of OpenNebula cloud management software with other open-source software ([4, 8]). However, there is little focus on the performance of the OpenNebula software in the existing literature. The performance evaluation aids new and existing research, industry and international projects when selecting OpenNebula software to their business. For example, if the duration for a small virtual machine (less number of CPU cores and RAM size is smaller) is less when compared to a big virtual machine (more number of CPU cores, RAM size is larger), the cloud consumers can select the small virtual machine to help heavily loaded virtual machine by sharing the load with a minimum duration of deployment.

**1.2 Problem Definition**

This thesis focuses on two major goals. The first goal is to develop a framework for comparing the features of the private-IaaS-cloud management software. The second goal is to evaluate the performance of the OpenNebula private-IaaS-cloud management software.
1.3 The Approach

To identify the features provided in different cloud management software, the following approach is taken. (1) Existing works related to the comparison of cloud management software are reviewed, as this study aids in selecting features for comparison. (2) Important features such as virtual machine management, etc, are selected for developing the framework. (3) Specific functionality related to the feature is selected. For example, specific virtual machine management operations, such as add, list, migrate, graphical interface and command-line interface are selected for the virtual machine management. Steps (1), (2) and (3) help to select the features for the comparison. (4) In order to analyze the functionality provided by the software, documents related to the software are reviewed. The review method is used to identify the availability of functionality. (5) The cloud management software is installed on the KTH LCN testbed for hands-on experience. Steps (4) and (5) are used to identify the features provided in the different cloud management software.

For the performance evaluation of the OpenNebula cloud management software, the following approach is taken. (1) Existing works are reviewed to identify evaluation metrics. (2) Virtual machine management operation is selected for the evaluation. (3) Selective functions, specifically adding, migrating and deleting are chosen for the evaluation. (4) Testing scenarios are identified for the evaluation such as different virtual machine types, number of virtual machines and change in the load of the system, as this reflects an operation environment. (5) The OpenNebula cloud management software is installed on the KTH LCN testbed for the performance evaluation.

1.4 Contribution of the Thesis

The first contribution is a framework for comparing features of various cloud management software. The features included in the comparison are physical host management, virtual machine management, virtual machine image management, virtual network management, user management, security, hypervisor, hybrid extension, quota, scheduling types, fault tolerance and cloud interface. The software included for comparisons are Eucalyptus, CloudForms, OpenStack, OpenNebula, Nimbus, openQRM, Abiquo and VMware vCloud. This framework can be used as a reference for evaluating other cloud management software. This allows cloud consumers to choose correct software based on their requirements. This allows the cloud management software vendors to include a missing feature in their software.

The second contribution is a performance evaluation of the OpenNebula cloud management software. The metrics used for evaluation are free memory, amount of network transfer and duration of operation. These metrics were measured for the virtual machine management operations. This clearly guides the cloud consumers to choose the specification of their physical servers and network equipment that satisfies their load requirements and to select migration type based on their requirements.
1.5 Outline of Thesis

The rest of this thesis is organized as follows. Related works for framework comparison and performance evaluation is discussed in Chapter 2. Chapter 3 discusses the available software and known methods that are used in this thesis. Chapter 4 explains the framework and the features of various cloud management software. Chapter 5 evaluates the performance of OpenNebula software with and without load. Chapter 6 summarizes this thesis with conclusion and further work.
2 Related Works

A holistic understanding of cloud computing related to Infrastructure-as-a-Service (IaaS) is discussed in [1] and [7]. The comparison of various cloud management software based on their features is discussed in [8], [9] and [11]. The benchmark that compares the performance of different cloud management software is discussed in [4], [10], [12] and [13].

The top ten obstacles and opportunities for growth in cloud computing are discussed in brief in [1]. They are service availability, data lock-in, data confidentiality, data transfer, performance issues, storage, large-scale bug fixing, scalability, reputation of cloud provider and software licensing. This paper [1] aids in selecting features for comparison.

The security features that are enhanced by the introduction of virtualization are discussed in [7]. This paper elaborates security vulnerabilities that can be exploited due to the misconfiguration of virtualization. In this thesis, the security features of virtualization are not discussed in detail, as different hypervisors provide different kind of virtualizations and the cloud management software supports many hypervisors. However, features such as security group and quota provided by different cloud management software are analyzed in this thesis, as they increase the security of the system. The main difference between the paper [7] and this thesis is that the paper [7] focuses on security at the hypervisor level whereas this thesis focuses on the security provided by cloud management software.

The architecture of the cloud management software, such as Eucalyptus, OpenNebula and Nimbus are discussed in paper [8]. Major focus is given to the features, such as hypervisor support, how users and administrator functions are separated, virtual machine management and virtual network management. In addition to these, this thesis discusses other features, such as quota, hybrid extension and scheduling. Other open-source cloud management software, such as OpenStack, openQRM and Abiquo are also included in the comparison. The main difference between the paper [8] and this thesis is that the paper [8] analyzed the features in detail (whether the DHCP service will run), whereas this thesis identifies the features in overview (whether the virtual network management is present or not).

Architecture, scheduling and communication between different components of the different open-source cloud management software, such as Xen cloud platform, Eucalyptus and OpenNebula are discussed in [9]. After discussing the basic architecture, a comparison is made for features, such as scheduling mechanism, available networking option, inter-component communication and interface provided for users and administrators. In this thesis both the open-source and commercial software are included in the framework for comparison. This helps to identify the features provided by the commercial software when compared to the open-source software.

Haizea, a resource release manager, can extend the scheduling of the cloud management software beyond immediate provisioning to best-effort releases and an advance reservation of
capacity is discussed in [11]. Haizea is used to lease hardware and software. Haizea provides three kinds of leases, advanced request (resource available at specific time), best-effort (resources are provided as soon as possible with the possibility of placing request in queue) and immediate lease (resources are provided on request). In this thesis, the various scheduling mechanisms that are available with cloud management software are discussed. No external software like Haizea is used for comparison as it is not possible to integrate Haizea in different cloud management software.

OpenNebula NFS eager and lazy disk allocation and Eucalyptus local disk eager allocation are covered extensively in [10]. The provisioning time of the virtual machine (1, 2, 4, 8, and 16 VMs) is measured for the different architectures. Performance of Wikipedia workload (throughput of web application) is evaluated for the different architectures. In this thesis, OpenNebula NFS eager disk allocation is used as it is a default configuration provided by the OpenNebula software. In this thesis, provisioning time for different virtual machine types, for different numbers of virtual machine and change in load of the system are evaluated, as this reflects an operating environment. This helps to identify whether the provisioning time increases linearly or exponentially. The main difference between the paper [10] and this thesis is that the paper [10] compares the performance between the different architectures whereas this thesis evaluates the performance of OpenNebula NFS eager disk allocation with the different virtual machine types, number of virtual machines and change in load of the system.

The verification of the amount of resource (RAM size, number of CPU cores and disk storage) assigned to a virtual machine based on the service level is discussed in [12]. Since the infrastructure is under the control of the cloud provider, the amount of these resources can be forged easily. The cloud users can verify these resources by themselves using the proof of work function. No verification for virtual machine’s resource is performed in this thesis, as this thesis focuses on a private cloud model, where the cloud provider provides a service to their business units.

Whether cloud computing can replace the need for super computers in research environment is discussed in [13]. Two sets of benchmark are run for performance evaluation, the HPC benchmark and the NAS parallel benchmark. These experiments are compared with a physical machine that matches EC2 instance, and their results are analyzed with an Amazon EC2 instance. In this thesis, Apache Olio is selected as a standard load, and the performance of OpenNebula software is evaluated with the Apache Olio load. The reason is to identify the performance variation in OpenNebula software as load increases and not in the context of replacing the super computer.

How the dynamic reconfiguration of the virtual machine (VM) affects the throughput of application running on a VM and the impact of migration to co-located VM’s are discussed in [4]. This paper explains the advantage of migration from the perspective of cloud users and cloud providers. In this thesis, migration characteristics such as migration time, maximum transfer rate
and the amount of data transfer are compared for different migration types and different virtual machine types. This thesis does not focus on performance of other parallel applications running in the system, as this experiment is conducted only to evaluate the migration characteristics change. The reason for analyzing the migration characteristics is that the migration operation increases the performance of the system by allowing the cloud provider to move the virtual machine from an overloaded physical machine to another physical machine.
3 Background

In this chapter the background of cloud computing, cloud management software and Zabbix software are discussed.

3.1 Cloud Computing

3.1.1 Definition of Cloud Computing

According to NIST, “Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [21].

3.1.2 Characteristics of Cloud Computing

According to NIST, the important characteristics of cloud comparing are discussed below [21].

Rapid elasticity is the capability of the system to rapidly scale upward or downward based on the demand. Measured service is the capability of the system to monitor, control and report resource usage (e.g., CPU, RAM, storage, Network bandwidth, etc.) and should provide metering capability for these resources. Broad network access is the capability of the system that can be reached over the network through a thin or a thick client platform (e.g., mobile phones, laptop, etc.). On-demand resource provisioning enables the customer to get required computing capabilities immediately (e.g., number of CPU cores, RAM size, etc.). Resource pooling is the capability of the system to provide location feature. Since the customer has no knowledge about the underlying infrastructure, some kind of location preference feature should be given on an abstract level (e.g., country, state or datacenter location) where the virtual machine will run.

3.1.3 Service Models

According to NIST, the service model can be classified into three types as discussed below [21].

Software-as-a-Service (SaaS) allows the consumers to directly use the application provided by the provider, using a thin interface (e.g., web browser). The consumers have no control over the infrastructure, operating system, network, storage and even the application they use. Google mail is a classical example for SaaS.

Platform-as-a-Service (PaaS) allows the consumers to create application using libraries, tools and services provided by the provider. Even though the consumers have no control over the infrastructure, they have full control of the deployed application. Google App Engine is a classical example for PaaS.
Infrastructure-as-a-Service (IaaS) allows the consumers to deploy any operating system and run any software. The consumers do not have control of the underlying cloud infrastructure, but they have full control of their part of virtual resources. Amazon EC2 is a classical example for IaaS.

3.1.4 Deployment Models

According to NIST, the deployment model can be classified into four types as discussed below [21].

In the public cloud model, the cloud infrastructure is provided by a cloud service provider for many customers. The customer leases the infrastructure provided and pays for the resource utilized.

In the private cloud model, the cloud infrastructure is provided by a single organization for its different business units. The infrastructure may exist on or off premises, and is maintained by the organization or third parties.

In the community cloud model, the cloud infrastructure is shared among customers with similar requirements (e.g., security policies). The management of a cloud environment can be handled by third parties.

In the hybrid cloud model, the cloud infrastructure provides extension for the private cloud to combine local resources with resources of the public cloud. The combination of the private and public cloud is the hybrid cloud.

3.2 Physical Machine

A physical machine is the combination of hardware and operating system. The hardware and operating system must be properly configured for running any application smoothly [36].

3.3 Virtual Machine

If the operating system runs on top of the hypervisor instead of the hardware, it is called the guest operating system or virtual machine, and if the operating system runs on top of the hardware, it is called the host operating system [7].

3.4 Network

Networking capabilities enable the underlying physical machines to communicate with each other. The network provides services such as DNS and DHCP. The DNS performs name resolution and the DHCP assigns IP address to the physical machines.

The network includes additional components, such as bridge-utils, for creating virtual bridge in the physical machine and for giving each virtual machine a unique MAC address.
The virtual networking is the process of assigning IP and MAC addresses to the virtual machines, which enables the virtual machines to communicate with each other. For more information about the basics of networking, see [37].

### 3.5 Virtual Machine Image

The virtual machine image consists of preconfigured operating system and software which is used to create a virtual machine immediately [22].

### 3.6 Hypervisor

The hypervisor is a software layer between the hardware and the operating system. This software layer allows the server hardware to be virtualized, so that multiple virtual machines can run on the same hardware.

The purpose of server virtualization is to increase server utilization by consolidating the virtual machines together in one or more physical servers. For more information about the hypervisor, see [14].

![Figure 1: Full virtualization (a) and hardware-layer virtualization (b) models [7]](image)

#### 3.6.1 Hypervisor Classification

Hypervisor can be classified into the following types.

If the hypervisor runs like a normal application on top of the host OS, as shown in figure 1.a, it is called full virtualization [7].

Para virtualization needs the guest OS to be modified in order to run on top of a hypervisor. The reason behind this is that the hypervisor model is simple and the performance is similar to non-virtualized hardware. Like full virtualization, the hypervisor will run on top of the host OS [7].
If the hypervisor runs directly on top of the hardware layer, as shown in figure 1.b, it is called hardware-layer virtualization.

### 3.6.2 Available Hypervisor

Major available hypervisors are listed below.

Xen uses the para virtualization technique. Here, the virtual machine is aware of the fact that it is running on top of a hypervisor and not directly on the server hardware [15].

KVM uses the full virtualization technique for the Linux-based OS. It allows the running of unmodified versions of Microsoft and Linux-based OS. The hardware must be capable to support virtualization extension, such as Intel VT or AMD-V [16].

Microsoft Hyper-V is the Microsoft’s hypervisor, dedicated to virtualize Microsoft-based OS [17].

VMware ESX uses the hardware-layer virtualization technique to run both Linux and Microsoft based OS [19].

### 3.7 Cloud Management Software

![Diagram of cloud management software interaction](image)

**Figure 2**: Interaction between the cloud management software and other components [7]
The cloud management software is a piece of software that orchestrates the whole process of adding the physical machine, getting the IP address lease from the network and creating the virtual machine. Figure 2 shows the interaction between the cloud management software and other components.

The first component ‘1’ is a physical machine, which is the combination of hardware and operating system. Cloud management software should be flexible enough to run on different operating systems and different vendors’ hardware. The second component ‘2’ is a network which includes DNS and DHCP. The network components should be designed to handle both the host and the virtual machine requests. The third component ‘3’ is the hypervisor which provides a framework to run virtual machines (VM). The fourth component ‘4’ is the image storage; this is where the VMs’ images are stored. The fifth component ‘5’ is the front-end which allows users to request the virtual machine and allows a cloud administrator to change configurations (e.g., adding more IP address to a DHCP server). The sixth component ‘6’ is the cloud management software (e.g., Eucalyptus, Open Nebula, Nimbus, etc.).

The cloud management software receives a request from the user and takes the corresponding VM’s image file to process it (e.g., adding a swap partition and padding the image to appropriate size) and sends a request to the hypervisor to create a virtual machine. Finally, the cloud management software requests a network component for IP and MAC assignment to the virtual machine. For more information regarding the cloud management software architecture, see [7].

3.8 Zabbix software

Zabbix is used to monitor the performance of a network and a server [20]. In this thesis, Zabbix is used to monitor CPU load, free memory used and the amount of network transfer. In this thesis, these parameters are called metrics. The Zabbix agent is installed on different machines where these metrics will be measured. The Zabbix master is installed on the machine, and the master is used to control and coordinate the Zabbix agent. The Zabbix agent sends these metrics’ values to the Zabbix master and the metrics are stored in the master database for the presentation as shown in figure 3.
3.9 SOAP and REST

SOAP stands for Simple Object Access Protocol. It uses XML format for passing messages, and it relies on an application layer protocol (e.g., HTTP, SMTP) for actual message transmission. For more information about the SOAP, see [2].

REST stands for Representational State Transfer. REST follows a traditional client and server architecture. For more information about the REST, see [113].
4 Framework for Comparing Features of Cloud Management Software

In this chapter a framework is created for comparing the features of various cloud management software. The framework consolidates the features provided by the different cloud management software and this allows the cloud consumers to choose software based on their business requirements. The software included for comparison are Eucalyptus, CloudForms, OpenStack, OpenNebula, Nimbus, openQRM, Abiquo and VMware vCloud.

4.1 Basic Information

Table 1 lists basic information about various cloud management software consisting of version used for developing the software and whether it is open-source or commercial software.

Table 1: Basic information about different cloud management software

<table>
<thead>
<tr>
<th>Cloud management software</th>
<th>Version</th>
<th>Open-source (O) / Commercial (C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eucalyptus community edition</td>
<td>2.0.3</td>
<td>O</td>
</tr>
<tr>
<td>Nimbus</td>
<td>2.7</td>
<td>O</td>
</tr>
<tr>
<td>CloudForms (Red Hat cloud)</td>
<td>2.2</td>
<td>C</td>
</tr>
<tr>
<td>OpenStack</td>
<td>Cactus release (0.2.1)</td>
<td>O</td>
</tr>
<tr>
<td>Abiquo community edition</td>
<td>1.7</td>
<td>O</td>
</tr>
<tr>
<td>openQRM</td>
<td>4.8</td>
<td>O</td>
</tr>
<tr>
<td>VMware vCloud (VMware cloud)</td>
<td>1.6</td>
<td>C</td>
</tr>
<tr>
<td>OpenNebula community edition</td>
<td>2.2</td>
<td>O</td>
</tr>
</tbody>
</table>

4.2 Software Architecture

The abstract software architecture of the cloud management software that is used for the framework comparison is explained in this section.

4.2.1 Cloud Architecture Presentation

Interface representation for the cloud management software architecture is shown in table 2.
Table 2: Interface representation of the cloud management software architecture

<table>
<thead>
<tr>
<th>Representation</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| ![Diagram](image) | • M1 represents the name of the physical server.  
• S1, S2… represent the software component running on top of the physical server. |
| ![Diagram](image) | Double-sided arrow indicates the network communication between the two servers. |
| ![Diagram](image) | Single-sided arrow indicates the communication between the user and the cloud management software for various management operation commands. |

### 4.2.2 Eucalyptus Architecture

The Eucalyptus architecture consists of five software components; they are Cloud Controller (CLC), Cluster Controller (CC), Walrus, Storage Controller (SC) and Node Controller (NC) as shown in figure 4.

There is a single CLC and Walrus for the entire cloud setup. The CLC provides an interface for a user to interact with the system and perform high-level cluster scheduling, such as identifying which cluster should be selected for creating a new virtual machine. The interface can be a normal web interface or SOAP interface or REST interface. See section 3.9 for more information about REST and SOAP.

Walrus is used to store virtual machine (VM) images. Walrus can be accessed through a SOAP or REST interface.

There will be a single CC and SC per cluster. The SC is used to store VM images within the cluster. The CC gathers information about a running VM and schedules a new VM to a particular NC.

The NC is responsible for controlling the hypervisor running on every physical machine. The NC performs virtual machine management operation.

For more information about the Eucalyptus cloud management software architecture, see [33].
4.2.3 Nimbus Architecture

The Nimbus architecture consists of three servers; they are the Service Node, the VMM Node and the Cloud client, as shown in figure 5.

The Service Node has the following components. The Nimbus IaaS performs services such as scheduling and virtual machine (VM) request. The Cumulus Storage is used to store virtual machine images.

The users have to download the Cloud client software, and the Cloud client software is required to interact with the Service Node. Using the Cloud client, users can upload the virtual machine images to the Cumulus storage and can instruct the Nimbus IaaS to start a virtual machine. Communication between the Cloud client and the Service Node takes place through HTTP.
VMM Node runs the hypervisor (Xen / KVM) and libvirt. For more information about the libvirt, see section 4.3.5. The hypervisor allows multiple virtual machines to run on top of it. The libvirt is used to manage (e.g., create, delete) a virtual machine running on top of the hypervisor.

![Diagram of Nimbus cloud management software](image)

Figure 5: The architecture of the Nimbus cloud management software [53]

Secure shell (SSH) is used for the communication between the Service Node and the VMM Node. DHCP server must be installed separately and DHCP daemon must be configured to provide IP address when the virtual machine boots up. For more information about Nimbus cloud management software architecture, see [53].

### 4.3.4 CloudForms Architecture

The CloudForms architecture consists of two servers; they are the CloudForms and the Worker node, as shown in figure 6.

![Diagram of CloudForms software components](image)

Figure 6: The software components of the CloudForms [110]

The cloud users use a cloud interface to interact with the system. The cloud interface has three components, namely the JBOSS ON, the Katello UI and the Aeolus UI component. The JBOSS ON component provides a function for monitoring and controlling the virtual machines. The Katello UI component provides an interface for interacting with the Katello component for the
image management. The Aeolus UI component provides an interface to perform the virtual machine management operations such as creating a new virtual machine. These three interfaces provide separated APIs for extension.

The Katello is used for building or modifying the virtual machine images. The images can be from Red Hat, Microsoft or user defined repositories and it is represented as Content in the figure 6.

The Conductor allows the cloud users to create a virtual machine that suits their requirements. An example is a bootable operating system with the specific software and the application specific configuration information, equivalent to contextualization. For more information about the contextualization, see section 4.3.11.

The Image Factory is used to create a new image with bootable operating system with the specific software and the application specific configuration. The created image will be stored in the Image Warehouse and can be used in the future.

The Condor provides a functionality to create, launch, monitor and remove the virtual machines. In addition to this, the Condor provides scheduling and enforces quota. The DeltaCloud provides an abstraction layer for the communication between the cloud user and the cloud provider. This allows the cloud users to use the same command to perform the cloud operations with the different cloud providers.

The Worker node runs the hypervisor and virtual machines are also run in these servers.

For more information about the CloudForms cloud management software architecture, see [110].

**4.2.5 OpenStack Cactus Architecture**

The OpenStack architecture consists of two servers; they are the Cloud controller and the Compute node, as shown in figure 7.

The Cloud controller has the following software components and these components can be installed in separate servers: nova-API, nova-schedule, nova-volume, nova-network, nova-database, queue, Swift and Glance. The nova-API allows a cloud consumer to interact with the system. The nova-API accepts the OpenStack API calls. The nova-schedule decides where to place a new virtual machine within the available physical machines. The nova-volume manages the creation, the attachment and the deletion of the persistent storage attached to the virtual machines. The nova-network performs the network management operations, such as creation, listing and deletion of the network. The nova-database stores information, such as the number of running virtual machines, available network options, list of users and list of projects. The queue will act like a hub in passing the information across these software components. The Glance provides an API to access the images and the Swift provides the storage for the virtual machine images.
The Compute node has a hypervisor and the nova-compute components. The hypervisor allows the multiple virtual machines to run on top of it. The nova-compute performs the virtual machine management operations, such as create, list and terminate a virtual machine.

![Diagram of OpenStack architecture](image)

**Figure 7:** The Architecture of the OpenStack cloud management software [65]

For more information about the OpenStack cloud management software architecture, see [65].

### 4.2.6 Abiquo Architecture

![Diagram of Abiquo architecture](image)

**Figure 8:** The Architecture of the Abiquo cloud management software [63]

Abiquo architecture consists of two servers; they are the abiCloud Server and the Worker node, as shown in figure 8.

The abiCloud Server consists of the following components. The Virtual Factory is used to manage the hypervisors (e.g., Xen, KVM). The Virtual System Monitor is used to manage the physical machines. The Storage System Manager is used to manage the persistent storage attached to the virtual machines. The Appliance Manager is used to manage the virtual machine images.

The Worker node runs a Hypervisor. The Storage system is used to store the virtual machine images.
The cloud users use a web interface to interact with the abiCloud Server. For more information about the Abiquo cloud management software architecture, see [63].

4.2.7 openQRM Architecture

Figure 9 shows the architecture of openQRM software. The openQRM consists of two servers; they are the openQRM-Server and the Worker node. The openQRM architecture is divided into the base and the plug-ins. The base provides a framework for the plug-in to interact and all the features are provided by its plug-in. The openQRM-Server provides the base for the following software, the Server-Management (to manage physical machines), the Storage-Management (to manage virtual machine images), the Deployment (to manage virtual machines), the Provisioning (to schedule a new virtual machine), and the High-Availability (to run openQRM service in two physical machines).

![openQRM Architecture Diagram]

Figure 9: The architecture of the openQRM cloud management software [111]

The ResourceAbstractionLayer provides the base for the hypervisor (KVM, Vmware ESX, XEN, etc) supported by the openQRM-Server software.

The StorageAbstractionLayer provides the base for the storage system (NetApp, NFS-Server, etc) supported by the openQRM-Server software.

The Worker node runs the hypervisor and virtual machines are also run in these servers. For more information on available plug-ins, see [113].

The openQRM-Server provides Network boot option for adding a new physical machine into the cluster through the network, instead of using the ISO images from the CD-ROM or the floppy disk. For more information on the network boot, see [112]. Limited documentation about the architecture and software component is published in their website.
4.2.8 VMware Architecture

The VMware architecture consists of three servers; they are the Virtual Infrastructure, the ESX Server host and the VirtualCenter Server, as shown in figure 10.

ESX Server host (hypervisor) virtualizes the server hardware so that multiple virtual machines can be run. For more information about the ESX server, see section 4.3.7.

The VirtualCenter server provides several management functions such as the physical machine management, the resource allocation, the virtual machine management, the virtual machine image management and the scheduling.

![Diagram](image)

Figure 10: The architecture of the VMware infrastructure [86]

The Virtual Infrastructure Client is used to configure the ESX Servers such as storage configuration, network configuration, used connect to the VM for console access and provides an interface to interact with the system.

The Windows version of Virtual Infrastructure Client is available to download. The web browser can be used to perform the virtual machine management operations through any networked system. For more information about the VMware architecture, see [86].

4.2.9 OpenNebula Architecture

The OpenNebula architecture consists of two servers; they are the FRONT-END and the CLUSTER NODES, as shown in figure 11.
FRONT-END has ONED, Drivers and Images components. The FRONT-END provides a command-line interface and this allows the cloud users to interact with the OpenNebula system. Drivers allow the users to plug in different storage, virtualization and monitoring techniques to the system. The Images represent the storage system where the virtual machine images are stored.

The ONED component is responsible for the operations such as scheduling, managing the virtual machines, managing the physical machines and managing the virtual network.

![Diagram of OpenNebula architecture]

Figure 11: The architecture of the OpenNebula cloud management software [107]

The CLUSTER NODE runs a hypervisor. The virtual machines are also run in these servers when requested. The communication between the FRONT-END and the CLUSTER NODE happens securely through an SSH connection.

For more information about the OpenNebula cloud management software architecture, see [107].

4.3 Framework

The elements that are included for creating a framework are physical host management, virtual machine management, virtual machine image management, virtual network management, user management, security, hypervisor, quota, scheduling types, hybrid extension, fault tolerance and cloud interface.

4.3.1 Physical Host Management

The first element that is included in the framework is the physical host management. The physical host management is the process of adding, listing and removing physical machines from the cloud management software. The reason for choosing this feature is that it allows the cloud provider to choose software that manage the physical machines. For more information about the physical machine, see section 3.2.

In addition to the physical host management, the user interface that is provided for the physical host management is also included in the framework. The user interface is a way through which
the cloud provider can interact with the cloud management software and perform the physical host management.

Table 3 shows the physical host management and the user interface provided by different cloud management software. ‘Y’ indicates that the feature is provided, ‘N’ indicates that the feature is not supported by the cloud management software, ‘G’ indicates that the cloud provider can use the graphical interface to manage the physical machines, and ‘C’ indicates that the cloud provider can use command-line interface to manage the physical machines.

Table 3: Physical host management for different cloud management software

<table>
<thead>
<tr>
<th>Type</th>
<th>Cloud management software</th>
<th>Physical host management</th>
<th>Graphical interface (G) / Command-line interface (C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-source</td>
<td>Eucalyptus community edition</td>
<td>Y [26]</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>Nimbus</td>
<td>Y [47]</td>
<td>G &amp; C</td>
</tr>
<tr>
<td></td>
<td>OpenStack</td>
<td>Y [57]</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>Abiquo community edition</td>
<td>Y [70, 71]</td>
<td>G &amp; C</td>
</tr>
<tr>
<td></td>
<td>openQRM</td>
<td>Y (Automatically detects physical server)[89]</td>
<td>G &amp; C</td>
</tr>
<tr>
<td>Commercial</td>
<td>VMware vCloud (VMware cloud)</td>
<td>Y [80]</td>
<td>G &amp; C</td>
</tr>
<tr>
<td></td>
<td>CloudForms (Red Hat cloud)</td>
<td>Y [110]</td>
<td>G</td>
</tr>
</tbody>
</table>

For openQRM see section 4.2.7, for more information about the automatic detection of the physical machines.

**4.3.2 Virtual Machine Management**

The second element that is included in the framework is the virtual machine management. The virtual machine management is the process of creating, listing, migrating (both live and cold migration) and deleting virtual machines from the cloud management software. The reason for
choosing this feature is that it allows the cloud consumers to choose software that manage the virtual machines (VM). For more information about the virtual machine, see section 3.4.

Table 4 shows the virtual machine management and the user interface provided by different cloud management software. ‘Y’ indicates that the feature is provided, ‘N’ indicates that the feature is not supported by the cloud management software, ‘G’ indicates that the cloud consumers can use the graphical interface to manage the virtual machine, and ‘C’ indicates that the cloud consumers can use the command-line interface to manage the virtual machine. In customization option, whether it is possible to create the virtual machine with different sizes (number of CPU cores and RAM size) is discussed. ‘Y’ indicates that the cloud consumers can create the virtual machine with different sizes and ‘N’ indicates that the cloud consumers can only create the virtual machine with a default size.

Table 4: Virtual machine management for different cloud management software

<table>
<thead>
<tr>
<th>Type</th>
<th>Cloud management software</th>
<th>Virtual machine management</th>
<th>Graphical interface (G) / Command-line interface (C)</th>
<th>Customization option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-source</td>
<td>Eucalyptus community edition</td>
<td>Y (except migrate) [27]</td>
<td>C</td>
<td>N</td>
</tr>
<tr>
<td>Nimbus</td>
<td>Y [48]</td>
<td>G &amp; C</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>OpenNebula community edition</td>
<td>Y [101,102,103]</td>
<td>G &amp; C</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>OpenStack</td>
<td>Y [58, 59]</td>
<td>G &amp; C</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Abiquo community edition</td>
<td>Y [70, 71]</td>
<td>G &amp; C</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>openQRM</td>
<td>Y [90,91]</td>
<td>G &amp; C</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Commercial</td>
<td>VMware vCloud (VMware cloud)</td>
<td>Y[80]</td>
<td>G &amp; C</td>
<td>Y</td>
</tr>
<tr>
<td>CloudForms (Red Hat cloud)</td>
<td>Y [110]</td>
<td>G</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>
The Eucalyptus software allows the cloud consumers to create the virtual machine with default size (number of CPU cores and RAM size) using the command-line interface and both the live and cold migration are not present.

The OpenStack software allows the cloud consumers to create the virtual machines with default size (number of CPU cores and RAM size) using the command-line interface and allows the cloud consumers to create the virtual machines with different sizes using the graphical interface.

The rest of the software allows the cloud consumers to create the virtual machines with different sizes using the provided interface.

### 4.3.3 Virtual Machine Image Management

Table 5: Virtual machine image management for different cloud management software

<table>
<thead>
<tr>
<th>Type</th>
<th>Cloud management software</th>
<th>Image management</th>
<th>Graphical interface (G) / Command-line interface (C)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Nimbus</td>
<td>Y [49]</td>
<td>G &amp; C</td>
</tr>
<tr>
<td></td>
<td>OpenStack</td>
<td>Y [60, 59]</td>
<td>G &amp; C</td>
</tr>
<tr>
<td></td>
<td>Abiquo community edition</td>
<td>Y [70, 71]</td>
<td>G &amp; C</td>
</tr>
<tr>
<td></td>
<td>openQRM</td>
<td>Y [91, 92]</td>
<td>G &amp; C</td>
</tr>
<tr>
<td>Commercial</td>
<td>VMware vCloud (VMware cloud)</td>
<td>Y [80, 81]</td>
<td>G &amp; C</td>
</tr>
<tr>
<td></td>
<td>CloudForms (Red Hat cloud)</td>
<td>Y [110]</td>
<td>G</td>
</tr>
</tbody>
</table>

The third element that is included in the framework is the virtual machine image management. The virtual machine image management is the process of adding, listing and deleting virtual machine images from the cloud management software. The reason for choosing this feature is that it allows the cloud consumers to choose software that manage the virtual machine images. For more information about the virtual machine image, see section 3.5.

Table 5 shows the virtual machine image management and the user interface provided by different cloud management software. ‘Y’ indicates that the feature is provided, ‘N’ indicates
that the feature is not supported by the cloud management software, ‘G’ indicates that the cloud consumers can use the graphical interface to manage the virtual machine images, and ‘C’ indicates that the cloud consumers can use the command-line interface to manage the virtual machine images.

All these software provide some standard virtual machine images to download from their websites, except Abiquo community edition and OpenNebula community edition. The standard images help to create virtual machines easily.

### 4.3.4 Virtual Network Management

Table 6: Virtual network management for different cloud management software

<table>
<thead>
<tr>
<th>Type</th>
<th>Cloud management software</th>
<th>Virtual network management</th>
<th>Static (S) / Dynamic (D)</th>
<th>Security group</th>
<th>Graphical interface (G) / Command-line interface (C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-source</td>
<td>Eucalyptus community edition</td>
<td>Y [31]</td>
<td>S &amp; D</td>
<td>Y</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>Nimbus</td>
<td>Y [50]</td>
<td>D</td>
<td>N</td>
<td>C &amp; G</td>
</tr>
<tr>
<td></td>
<td>OpenStack</td>
<td>Y [61]</td>
<td>S &amp; D</td>
<td>Y</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>Abiquo community edition</td>
<td>Y [70, 71]</td>
<td>S &amp; D</td>
<td>N</td>
<td>G &amp; C</td>
</tr>
<tr>
<td></td>
<td>openQRM</td>
<td>Y [91, 93]</td>
<td>S &amp; D</td>
<td>N</td>
<td>G &amp; C</td>
</tr>
<tr>
<td>Commercial</td>
<td>VMware vCloud (VMware cloud)</td>
<td>Y [78, 80]</td>
<td>S &amp; D</td>
<td>Y</td>
<td>G &amp; C</td>
</tr>
<tr>
<td></td>
<td>CloudForms (Red Hat cloud)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The fourth element that is included in the framework is the virtual network management. The virtual network management is the process of adding, listing, and removing virtual networks from the cloud management software. The reason for choosing this feature is that it allows the cloud providers to choose software that manage the virtual networks. For more information about the virtual networks, see section 3.3.

The virtual network can be static or dynamic. In a static virtual network, an IP address and MAC pairs are given in a static file and the cloud management software assigns an IP address to VM
from this file. In a dynamic virtual network, the IP address to VM is assigned directly by the DHCP server. The reason for identifying this feature is that it allows the cloud providers to choose software that allows configuring the DHCP server to give an IP address to VM from a subnet (e.g., /20) instead of creating a file with 5000 static entries for 5000 VMs.

The security group is a group of virtual machines that can share data among themselves. In security group, the presence (or not) of an inbuilt mechanism for filtering the network traffic (inbound traffic / outbound traffic) is analyzed for the cloud management software. The reason for identifying this feature is that it allows the cloud providers to choose software that provides a fine-grained traffic control. For example, a DB server can accept traffic only from a web server and not from others.

Table 6 shows the virtual network management, the static or dynamic configuration, the security group and the user interface provided by different cloud management software. ‘Y’ indicates that the feature is provided, ‘N’ indicates that the feature is not supported by the cloud management software, ‘G’ indicates that the cloud provider can use the graphical interface to manage the virtual networks, ‘C’ indicates that the cloud provider can use the command-line interface to manage the virtual networks, ‘S’ indicates the presence of static virtual network configuration, ‘D’ indicates the presence of dynamic virtual network configuration and blank indicates that the information could not be found from the website.

**4.3.5 Cloud Interface**

The fifth element that is included in the framework is the cloud interface. The cloud interface is a way to administer the virtual machine images, virtual networks and virtual machines. If cloud users have an experience with a specific cloud interface (e.g., Amazon / Libvirt) they can use the known cloud interface to interact with the software instead of learning a new interface. The reason for choosing this feature is that it allows the cloud users to choose software that provide support to the familiar cloud interface. Table 7 shows the list of cloud interfaces provided by the software such as proprietary API, Amazon EC2 API, Libvirt API and OCCI API.

The proprietary API is specific to cloud management software and it cannot be used by any other cloud management software.

The Amazon EC2 API provides the functionality exposed by the EC2 API and the API can be used with different cloud management software. The EC2 API provides the functionality for performing the virtual machine management operations (see the column titled “virtual machine management” in table 4) [43].

The Libvirt API provides the functionality for performing both the virtual machine management operations (see the column titled “virtual machine management” in table 4) and the virtual network management operations (see the column titled “virtual network management” in table 6). The libvirt allows an interaction with the recent version of Linux-based OS [44].
Table 7: Cloud interface provided by different cloud management software

<table>
<thead>
<tr>
<th>Type</th>
<th>Cloud management software</th>
<th>Cloud interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-source</td>
<td>Eucalyptus community edition</td>
<td>Amazon EC2 API, Proprietary API, OCCI API [25, 38]</td>
</tr>
<tr>
<td></td>
<td>Nimbus</td>
<td>Amazon EC2 API, Proprietary API, Libvirt API [46, 18]</td>
</tr>
<tr>
<td></td>
<td>OpenNebula community edition</td>
<td>OCCI API, Amazon EC2 API, Libvirt API, Proprietary API [99]</td>
</tr>
<tr>
<td></td>
<td>OpenStack</td>
<td>Amazon EC2 API, Proprietary API, OCCI API [56, 38]</td>
</tr>
<tr>
<td></td>
<td>Abiquo community edition</td>
<td>Proprietary API [69]</td>
</tr>
<tr>
<td></td>
<td>openQRM</td>
<td>Proprietary API [88]</td>
</tr>
<tr>
<td>Commercial</td>
<td>VMware vCloud (VMware cloud)</td>
<td>Proprietary API, Libvirt API [77, 18]</td>
</tr>
<tr>
<td></td>
<td>CloudForms (Red Hat cloud)</td>
<td>Proprietary API, Libvirt API [110, 18]</td>
</tr>
</tbody>
</table>

The OCCI API provides a general API so that it can be integrated with different cloud management software. The OCCI API provides the functionality for performing both the virtual machine management operations (see the column titled “virtual machine management” in table 4) and the virtual network management operations (see the column titled “virtual network management” in table 6) [45].

4.3.6 User Management

The sixth element that is included in the framework is the user management. The cloud providers are administrators who setup the cloud infrastructure and the cloud users use the infrastructure provided by the administrator to complete their tasks. The user management is the process of adding, monitoring and removing users from the system. The reason for choosing this feature is that it allows the cloud providers to choose software that manage the cloud users.

A number of individual who are gathered or organized to perform a similar function is called as a group (such as manufacturing department, HR department). The configuration of new group apart from the cloud providers and cloud users is discussed in this part of the framework. The reason for choosing this feature is that it allows the cloud providers to choose software that gives the possibility to add a new group.
Table 8 shows the user management and the user interface provided by different cloud management software. ‘Y’ indicates that the feature is provided, ‘N’ indicates that the feature is not supported by the cloud management software, ‘G’ indicates that the cloud provider can use the graphical interface to manage the cloud users, ‘C’ indicates that the cloud provider can use the command-line interface to manage the cloud users, ‘A’ indicates that the cloud provider can configure a new group, and ‘X’ indicates that the new group cannot be configured.

Table 8: User management for different cloud management software

<table>
<thead>
<tr>
<th>Type</th>
<th>Cloud management software</th>
<th>User Management</th>
<th>Graphical interface (G) / Command-line interface (C)</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Nimbus</td>
<td>Y [52]</td>
<td>G &amp; C</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>OpenStack</td>
<td>Y [62]</td>
<td>C</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Abiquo community edition</td>
<td>Y [70, 71, 73]</td>
<td>G &amp; C</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>openQRM</td>
<td>Y [95, 91]</td>
<td>G &amp; C</td>
<td>A</td>
</tr>
<tr>
<td>Commercial</td>
<td>VMware vCloud (VMware cloud)</td>
<td>Y [80, 82]</td>
<td>G &amp; C</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>CloudForms (Red Hat cloud)</td>
<td>Y [110]</td>
<td>G</td>
<td>A</td>
</tr>
</tbody>
</table>

4.3.7 Hypervisor

The seventh element that is included in the framework is a hypervisor. The hypervisor or virtual machine monitor (VMM) allows hardware to be virtualized, so that multiple guest operating systems can run on the same hardware. For more information about the hypervisor, see section 3.6. The reason for choosing this feature is that it allows the cloud providers to choose software that provides a specific hypervisor.

Table 9 lists the hypervisors supported by the different cloud management software; ‘Y’ indicates that the hypervisor model is supported and ‘N’ indicates that the hypervisor model is not supported.
Table 9: Hypervisor supported by different cloud management software

<table>
<thead>
<tr>
<th>Type</th>
<th>Cloud management software</th>
<th>Xen</th>
<th>KVM</th>
<th>Microsoft Hyper –V</th>
<th>VMware ESX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-source</td>
<td>Eucalyptus community edition</td>
<td>Y [34]</td>
<td>Y [34]</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Nimbus</td>
<td>Y [54]</td>
<td>Y [54]</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Abiquo community edition</td>
<td>Y [75]</td>
<td>Y [75]</td>
<td>N</td>
<td>Y [75]</td>
</tr>
<tr>
<td></td>
<td>openQRM</td>
<td>Y [96]</td>
<td>Y [96]</td>
<td>N</td>
<td>Y [96]</td>
</tr>
<tr>
<td>Commercial</td>
<td>VMware vCloud (VMware cloud)</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y [85]</td>
</tr>
</tbody>
</table>

4.3.8 Scheduler

The eighth element that is included in the framework is a scheduler. The scheduler decides which physical machine to place a new virtual machine (VM), within the available physical machines. The reason for selecting the scheduler is to choose the software based on the following features.

The capacity is the number of CPU cores and RAM size that is available in a physical machine (e.g., 24 CPU cores and 64GB of RAM).

The power management feature is consolidating the VMs by using fewer physical machines. The power management feature is used to increase the efficiency of energy consumption. When the configuration is set to run in the static mode, the physical machine with the maximum number of running VMs at that moment is selected for creating a new VM. When the configuration is set to run in the dynamic mode, the power conservation is achieved by a migration of VMs from the physical machines that has not reached its capacity to the machines that can attain the maximum capacity.

The load balance feature identifies the physical machine with the least load (number of unused CPU cores is the highest) for running a VM. The load balance feature increases the overall performance of the system. When the configuration is set to run in the static mode, the physical machine with the least load at that moment is selected for creating a new VM. When the configuration is set to run in the dynamic mode, the load balancer migrates virtual machines evenly across the system by assessing the current load across the physical machines.
The resource pool can map the allocation of the physical resources such as the number of CPU cores and the RAM size to an organizational unit (e.g., HR, development, testing, etc). This allows the delegation of the administrative control to the respective organizational units. For example, HR group is assigned ten CPU cores and 20 GB of RAM, and full administration access to these resources is given to the HR group.

The co-location of VMs is running a group of VMs on the same physical machine (for example, if the web server has to access the application server for getting the web pages, they can be placed on the same physical machine). The anti-co-location of VMs is not allowing a pair of VMs to run on a single physical machine (for example, the primary and back-up web server VMs should always run on different physical machines).

Table 10 shows the scheduling provided by the different cloud management software. ‘Y’ indicates that the feature is provided, ‘N’ indicates that the feature is not supported by the cloud management software, ‘S’ indicates that the feature is supported in a static mode in the power management and the load balance, ‘D’ indicates that the feature is supported in a dynamic mode in the power management and the load balance, and the blank indicates the undetermined mode since the information cannot be found from the website of the software provider.

The default scheduling mechanisms that are available with the different cloud management software is listed below.
The round-robin scheduler will choose the physical machine in a circular order for creating a new VM. For example, assuming two machines, the first VM will be placed in the machine 1, the second VM will be placed in the machine 2, and the third VM will be placed once again in the machine 1 [35].

The Eucalyptus community edition has two scheduling mechanisms; they are greedy scheduler and round-robin scheduler. The greedy scheduler will choose the physical machine with the least number of running VMs for creating a new VM [35].

The Nimbus has two scheduling mechanisms; they are round-robin scheduler with extension and greedy scheduler. The round-robin scheduler with extension will choose the physical machine that has the highest free memory (RAM size) for creating a new VM. The greedy scheduler will choose the physical machine that has the least free memory (RAM size) for creating a new VM. This is a static power management – the free memory on the physical machine keeps on decreasing as the VMs are created [55].

The CloudForms uses Condor as the scheduler. The Condor provides the round-robin scheduler. The Condor can pack the virtual machines with limited physical machines (static power management). For example, assuming two machines, the first VM will be placed in the machine 1, the second VM will be placed in the machine 1. If there are not enough resources in the machine 1, the third VM will be placed in the machine 2. The Condor reserves the resources (e.g., number of CPU cores, RAM size, etc) in advance (specific date and time) for the business critical applications, such as payroll administration [110].

The OpenStack has three scheduling mechanisms; they are simple scheduler, chance scheduler and availability zone scheduler. The simple scheduler uses the load balancer to identify the physical machine with the least loads (the number of unused CPU cores is the highest) for creating a new VM (static load balance). A cluster is a group of physical machines connected together and it can be logically viewed as a single physical machine. The chance scheduler will choose the physical machine with enough resources randomly for creating a new VM (physical machine will be selected from any available cluster). The availability zone scheduler will choose the physical machine with enough resources randomly within the given cluster for creating a new VM (physical machine will be selected from the specific cluster) [67].

The Abiquo community edition has a best-fit scheduler. The physical machine that has the highest number of unused CPU cores is selected (static load balance). If two physical machine matches (the number of unused CPU cores is same), the physical machine with less capacity is selected. For example, assuming two machines, the machine installed with 16 CPU cores is selected when compared to the machine installed with 30 CPU cores [68].

For openQRM software, information about scheduler could not be found from the website.

The VMware vCloud uses a distributed resource scheduler (DRS) and the DRS has the following features. The first feature is maintenance mode. When the physical machine is set to the
maintenance mode, the DRS will automatically live migrate the running virtual machines (VMs) from the source physical machine to other available physical machines. The second feature is resource pool. The DRS will combine resources of the physical machines (e.g., number of CPU cores, RAM size) as a resource pool and allows compartmentalizing this resource pool into an organizational structure (e.g., HR, development, testing, etc). The third feature is automatic and manual mode. The DRS will place a new VM in one of the available physical machines. When DRS is configured in the automatic mode, it migrates the running VMs to another physical machine based on the load condition (number of unused CPU cores) of the physical machine (dynamic load balance). When DRS is configured in the manual mode, it will only provide suggestion for the better performance. The fourth feature is resource reservation. The DRS reserves the resources (e.g., number of CPU cores, RAM size, etc) in advance (specific date and time) for the business critical applications, such as payroll administration. The fifth feature is co-location and anti-co-location of VMs. It is possible for the DRS to place a group of VMs on a fixed physical server (co-location of VMs). For example, if the web server VM has to access the application server VM for getting web pages, they can be placed on the same physical machine and this reduces a lot of network traffic. It is possible for the DRS not to place a pair of VMs on a fixed physical server (anti-co-location of VMs). For example, the primary and back-up VMs (e.g., web server) should always run on different physical machines. The sixth feature is power management. The DRS can increase the efficiency of energy consumption by consolidating the virtual machines together and turning off unused physical servers (dynamic power management) [86].

The OpenNebula community edition has three scheduling mechanisms; they are packing policy, striping policy, and load-aware policy. The packing policy will choose physical machine with the highest number of running VMs for creating a new VM (static power management). The striping policy will choose the physical machine with the least number of running VMs for creating a new VM. The load-aware policy will choose the physical machine with the number of unused CPU cores is highest for creating a new VM (static load balance) [109].

4.3.9 Quota

The ninth element that is included in the framework is a quota. The quota provides the ability to impose the limitation for the virtual machine requests. The reason for choosing this feature is that it allows the cloud providers to choose software that provides a fair play among the cloud users, such that no single user can consume all resources. The resources that are included for comparison are listed below.

The possibility of controlling the resources per user request (e.g., user A can use maximum of two CPU cores when creating a virtual machine) or per group usage (e.g., users registered with the HR group can run the maximum of 10 virtual machines altogether) is discussed.
The evaluation criteria used are the limitations of the number of CPU cores, the RAM size, the public IPs that can be allocated to the virtual machine or a group of virtual machines and the number of virtual machines that a user can run simultaneously for the chosen software.

Table 11: Quota provided by different cloud management software

<table>
<thead>
<tr>
<th>Type</th>
<th>Cloud management software</th>
<th>User (U) / Group (G)</th>
<th>CPU</th>
<th>RAM</th>
<th>Public IP</th>
<th>Number of virtual machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-source</td>
<td>Eucalyptus community edition</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Nimbus</td>
<td>U[51]</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>OpenNebula community edition</td>
<td>U[106]</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y[106]</td>
</tr>
<tr>
<td></td>
<td>OpenStack</td>
<td>U &amp; G [64]</td>
<td>Y</td>
<td>N</td>
<td>Y[64]</td>
<td>Y[64]</td>
</tr>
<tr>
<td></td>
<td>Abiquo community edition</td>
<td>U &amp; G [74]</td>
<td>Y</td>
<td>Y</td>
<td>Y[74]</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>openQRM</td>
<td>U &amp; G [95]</td>
<td>Y</td>
<td>Y</td>
<td>Y[95]</td>
<td>Y[95]</td>
</tr>
<tr>
<td></td>
<td>CloudForms (Red Hat cloud)</td>
<td>U &amp; G [110]</td>
<td>Y</td>
<td>Y</td>
<td>Y[110]</td>
<td>Y[110]</td>
</tr>
</tbody>
</table>

Table 11 shows the quota provided by different cloud management software. ‘Y’ indicates that the limitation is provided for the CPU, RAM, number of virtual machines and public IP usage; ‘N’ indicates that the limitation is not provided by the software; ‘U’ indicates that the limitation is based on per user request, and ‘G’ indicates that the limitation is based on the group usage.

4.3.10 Hybrid Extension

The tenth element that is included in the framework is a hybrid extension. The hybrid extension is the process of providing an extension for a private cloud to combine the local resources with the resources from a public cloud. The reason for choosing this feature is that it allows the cloud consumers to choose software that provides the hybrid extension [23].

Table 12 lists the hybrid extensions provisioned in the software. ‘Y’ indicates that the feature is provided and ‘N’ indicates that the feature is not provided.
Table 12: Hybrid Support provided by different cloud management software

<table>
<thead>
<tr>
<th>Type</th>
<th>Cloud management software</th>
<th>Hybrid extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-source</td>
<td>Eucalyptus community edition</td>
<td>Y [24]</td>
</tr>
<tr>
<td></td>
<td>Nimbus</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>OpenNebula community edition</td>
<td>Y [97]</td>
</tr>
<tr>
<td></td>
<td>OpenStack</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Abiquo community edition</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>openQRM</td>
<td>Y [87]</td>
</tr>
<tr>
<td>Commercial</td>
<td>VMware vCloud (VMware cloud)</td>
<td>Y [76]</td>
</tr>
<tr>
<td></td>
<td>CloudForms (Red Hat cloud)</td>
<td>Y [111]</td>
</tr>
</tbody>
</table>

Further investigation is needed to identify the list of public clouds that the cloud management software can extend and to what extent.

4.3.11 Contextualization

Table 13: Contextualization supported by different cloud management software

<table>
<thead>
<tr>
<th>Type</th>
<th>Cloud management software</th>
<th>Contextualization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-source</td>
<td>Eucalyptus community edition</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Nimbus</td>
<td>Y [42]</td>
</tr>
<tr>
<td></td>
<td>OpenNebula community edition</td>
<td>Y [98]</td>
</tr>
<tr>
<td></td>
<td>OpenStack</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Abiquo community edition</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>openQRM</td>
<td>N</td>
</tr>
<tr>
<td>Commercial</td>
<td>VMware vCloud (VMware cloud)</td>
<td>Y [79]</td>
</tr>
<tr>
<td></td>
<td>CloudForms (Red Hat cloud)</td>
<td>Y [110]</td>
</tr>
</tbody>
</table>

The eleventh element that is included in the framework is a contextualization. The contextualization is the process of adding a specific configuration to a virtual machine at the time of creation. Some of the examples of contextualization are assigning a specific IP address to the virtual machine, adding static files to the web server, adding a network attached storage to the virtual machine. The reason for choosing this feature is that it allows the cloud providers to choose software that provides the contextualization.
Contextualization support provided by the different cloud management software is listed in table 13. ‘Y’ indicates that the feature is provided and ‘N’ indicates that the feature is not provided.

Further investigation is needed to identify how this feature is provided. For example, using XML format or running scripts at the time of creating the virtual machines.

4.3.12 Fault Tolerance
The twelfth element that is included in the framework is a fault tolerance. The fault tolerance is the process that provides a failover mechanism to the cloud management software even in case of an accidental software component failure. The reason for choosing this feature is that it allows the cloud providers to choose software that provides a high availability.

The Eucalyptus community edition has a single point of failure, if the Cloud Controller fails the system is isolated, see section 4.2.2 [33]. The Nimbus has a single point of failure, if the Nimbus IaaS service fails the system is isolated, see section 4.2.3 [53]. The CloudForms has a single point of failure, if the Conductor fails the system is isolated, see section 4.2.4 [110]. The OpenStack has a single point of failure, if any of the nova service fails the system is isolated, see section 4.2.5 [65]. The Abiquo community edition has a single point of failure, if any of the abicloud Server components fails the system is isolated, see section 4.2.6 [113]. The openQRM has a single point of failure, if any of the openQRM-Server components fails the system is isolated, see section 4.2.7 [111]. The VMware vCloud has a single point of failure, if the VirtualCenter Server fails the system is isolated, see section 4.2.8 [84]. The OpenNebula community edition has a single point of failure, if the OpenNebula daemon fails the system is isolated, see section 4.2.9 [107].

To sum up, all the software considered for this study has a single point of failure. Further investigation is needed to identify the impact of every software component failure that is shown in architecture.

4.4 Overall Conclusion
A framework is developed for comparing features of various cloud management software. This allows cloud customers to choose software based on their requirements. This framework helps other cloud management software vendors to add certain features that are missing when compared to other software. This framework can be taken as a reference for developing new cloud management software.

4.4.1 Major Contribution
The major contribution related to the framework is identifying features provided for the commercial software that are not available for the open-source software. The features are: (1) co-location of VMs is running a group of VMs on the same physical machine (for example, if the web server VM has to access the application server VM for getting the web pages, they can be
placed on the same physical machine); (2) anti-co-location of VMs is not allowing a pair of VMs to run on a single physical machine (for example, the primary and back-up web server VMs should always run on different physical machines); (3) the resources of the physical machines can be combined (e.g., number of CPU cores, physical memory) as a resource pool and compartmentalized into an organizational structure (e.g., HR, development, testing, etc).
5. Performance Evaluation of OpenNebula Cloud Management Software

This section evaluates the performance of the OpenNebula cloud management software in no-load and load setups. Specifically, the free memory and amount of network transfer for different virtual machine management operations, such as add, delete, live migrate and cold migrate are evaluated. Physical host management and virtual network management operations are not shown in the results, since the end time of these operations have not been measured.

5.1 Testbed Description

5.1.1 Testbed Description for No-Load Setup

Figure 12 illustrates the testbed for a no-load setup. The full virtualization (see section 3.6) technique is used here, where the hypervisor runs as a normal application (e.g., SSH, NFS, etc.) on top of the server OS. The Hypervisor provides an abstraction layer for the underlying hardware, and this allows the running of multiple operating systems in parallel. The Multiple Guest OS Ubuntu 10.04 LTS is the virtual machine (see section 3.4) running on top of the hypervisor. In this thesis, Multiple Guest OS Ubuntu 10.04 LTS is referred to as a virtual machine (VM). The chosen server OS is Ubuntu server 10.04 LTS (64 bit), because of its long-term support from the open-source community. In this thesis, Ubuntu server 10.04 LTS is referred to as the Base OS. The chosen Hypervisor is KVM because it is easy to install KVM on Base OS. All servers are connected via a 1Gbps switch.

Server 3 is the Cloud Master where the OpenNebula daemon (oned process) runs. The oned process is required to allow an interaction between users and the cloud management software.
The Cloud Operation Code required to perform a cloud management operation has been developed in Java. This code is written as part of this thesis work.

Server 1 and Server 2 are Cloud Slaves, where the hypervisor runs. Virtual machines are also run in these servers.

5.1.2 Testbed Description for Load Setup

Figure 13 illustrates the testbed for a load setup. This setup is similar to the no-load setup, with the following additional components:

- Apache Olio, a social networking application, is used to manage events [40]. Olio runs on a VM hosted by Server 2.
- Server 4, runs Faban (10 threads) [41], a workload generator, to drive the load against the Olio application with 5000 concurrent users.

Installation and set up of Olio PHP can be performed using [74] and set up of Faban can be performed using [95].

![Diagram of testbed for load setup](image)

Figure 13: Testbed for load setup

5.2 Hardware Specification

All servers share the same specifications as listed below,

1. Hardware name: Dell PowerEdge R715 2U Rack Server
2. CPU Specification:
   - 24 core processors and each core has the following specification,
   - CPU model name: AMD Opteron(tm) Processor 6172
   - CPU frequency: 2100.246 MHz
   - CPU cache size: 512 KB
3. RAM: 64GB
4. Disk size: 500GB (single disk)
5. NIC card: Two Dual-Port Broadcom 5709C Gigabit NICs (Total of 4x 1GbE ports)

5.3 Software Specification

OpenNebula requires the software listed in Table 14. The Base OS of all servers is the Ubuntu 10.04 LTS (64 bit) server edition.

Table 14: List of software required to run OpenNebula

<table>
<thead>
<tr>
<th>Cloud Master (Server 3)</th>
<th>Cloud Slave (Server 1 &amp; Server 2)</th>
</tr>
</thead>
</table>
| ruby 1.8.6, sqlite 3.5.2, libxmlrpcc 1.0.6, openssl 0.9, libsll3-dev, libxmlrpcc3-dev, libssl-dev, scons 0.9.7, g++ 4, flex 2.5, bison 2.3, libxml2-dev, ruby-dev, rubygems, rake, make, ruby xmlparser, ruby nokogiri and OpenNebula 2.2 | SSH server
qemu-kvm
libvirt-bin
ubuntu-vm-builder
bridge-utils
ruby 1.8.5 |

5.4 Interaction between Components

The Cloud Operation Code is written in Java, using the OpenNebula Java API (see appendix A) to perform cloud management operation, such as virtual machine management (see section 4.3.2). Oneadmin user account (Linux-user account “Oneadmin”) is created on all three servers, after installing Base OS. Password-less SSH access needs to be given across the three servers for the Oneadmin user account.

OpenNebula uses a shared file system on Cloud Master (Server 3) for storing VM images. For this purpose, NFS is created during installation and read/write accesses are given to the Oneadmin user account on Server 1 and Server 2.

When the Cloud Operation Code is issued on Server 3, oned process on Server 3 is invoked to execute the Cloud Operation Code. The oned process on Server 3 creates a template file based on the underlying Hypervisor. The created template file is placed on the NFS by the oned process, so that these files can be accessed by other servers (Server 1 & Server 2). Oned process on Server 3 uses a SSH connection to invoke the corresponding cloud management operations on Server 1 or Server 2.

For example, in live migration of a virtual machine from Server 1 to Server 2, a live migration command is issued on Server 3 using the OpenNebula Java API. The oned process on Server 3 is invoked and this in-turn instructs the Hypervisor on Server 1 to perform a live migration of a VM from Server 1 to Server 2. If the migration is successful, the oned process receives a
response from the Server 2’s hypervisor and updates the state information on the Server 3’s database. If the migration is unsuccessful, state information is not updated in Server 3’s database.

Figure 14 illustrates the interaction between the Faban and Apache Olio where the Faban drives load against the Apache Olio.

![Figure 14: Faban Setup](image)

5.5 Evaluation Metrics

The metrics listed in Table 15 are used for performance evaluation. These metrics are measured for every second. The experiments are performed once and the results are reported.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration of Operation [sec]</td>
<td>The start time of the cloud management operation is obtained from Cloud Operation Code, as it triggers the operation on Server 3. The stop time is obtained from a log file presented on Server 3 (/var/log/one/oned.log). The time difference between start time and stop time is measured in seconds.</td>
</tr>
<tr>
<td>Free Memory [GB]</td>
<td>It is the amount of RAM that is unused or freely available on a server.</td>
</tr>
<tr>
<td>Amount of network Transfer [MB/s]</td>
<td>Indicates how much data is transferred between two servers in one direction (network transmit) to perform a cloud management operation.</td>
</tr>
</tbody>
</table>

5.6 Virtual Machine Description

In this thesis, two different types of virtual machine are considered for the performance evaluation. They are: (1) Small-Instance Virtual Machine (SIVM) has 1 CPU core, 1.7 GB of Memory and is installed with Ubuntu server 10.04 LTS (64 bit); (2) Large-Instance Virtual Machine (LIVM) has 2 CPU cores, 7.5 GB of Memory and is installed with Ubuntu server 10.04 LTS (64 bit).
5.7 Virtual Machine Management

Commands listed in Table 16 are considered for the performance evaluation.

Table 16: Virtual machine management commands executed on OpenNebula

<table>
<thead>
<tr>
<th>Command</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| **Add VM on Server X**           | Add VM includes the following steps,  
|                                  | - Cloning: a specific image file is cloned on Server 3.  
|                                  | - File creation: the directory structure is created with the proper file and  
|                                  | folder permission in NFS on Server 3.  
|                                  | - Copying image file: the cloned image file is copied to the newly  
|                                  | created directory on Server 3  
|                                  | - Executing VM: the VM will be executed using the corresponding  
|                                  | image file on Server X. |
| **Delete VM on Server X**        | Delete VM will terminate a VM ungracefully, by losing all state information  
|                                  | stored in RAM. The command deletes all files and folders allocated to the  
|                                  | VM on Server 3 and frees up resources (CPU, RAM) on Server X. |
| **Cold Migrate VM from Server X to Server Y** | The command performs a cold migration. The command stops a VM and  
|                                  | saves the states of all processes and applications running in the VM in a  
|                                  | separated file on Server X (state file). Then, the VM’s data along with the  
|                                  | state file is transferred from Server X to Server 3. After that, Server 3  
|                                  | transfers the VM’s data along with the state file to Server Y. |
| **Live Migrate VM from Server X to Server Y** | The command performs a live migration. The command transfers the VM to  
|                                  | the Server Y without halting on the Server X during the migration. Live  
|                                  | migration is performed in three phases:  
|                                  | - Warm-up phase: In the first phase, all memory pages of VM on Server  
|                                  | X are transferred to Server Y  
|                                  | - Subsequent warm-up phase: This phase is repeated multiple times. Every time, memory pages that are dirtied during the previous time on  
|                                  | Server X are transferred to Server Y.  
|                                  | - Stop and copy phase: In this phase, VM is stopped on Server X and  
|                                  | remaining dirty pages are copied to Server Y. |

5.8 Scenario Results

The following experiments evaluate the OpenNebula cloud management software for virtual machine management operations, consisting of add, delete, live migrate and cold migrate.
5.8.1 Evaluating Free Memory for Adding Virtual Machines in the No-load Setup

Operation

For adding 2 SIVMs, one SIVM is added to Server 1 while the other SIVM is added to Server 2. The actual memory consumed on the Cloud Slave for a SIVM is around 0.19GB, measured using the Linux command (free) during a trial test. The Ubuntu 10.04 LTS image file (with SSH access) stored on the Cloud Master is approximately 11GB, measured using the Linux command (du).

![Figure 15: Free memory for adding 2 SIVMs in the no-load setup](image)

Observation

Figure 15 shows the free memory for adding 2 SIVMs in the no-load setup. The duration for adding 2 SIVMs is 142 seconds.

Initially, the free memory on Server 3 is constant at 51.52GB (from 6s to 24s), and then there occurs a linear decrease in free memory from 51.52GB at 24s to 39.09GB at 36s followed by another linear decrease in free memory from 39.09GB at 36s to 30.94GB at 135s and once again the free memory becomes constant 30.94GB (from 135s to 148s).

The free memory on Server 1 is constant, 60.02GB (from 6s to 146s), and then there occurs a linear decrease from 60.02GB at 146s to 59.52GB at 148s. The free memory on Server 2 is constant, 58.88GB (from 6s to 145s), and then there occurs a linear decrease from 58.88GB at 145s to 58.37GB at 148s.
Discussion and Conclusion

The free-memory reduction on Server 1 is 0.50GB (from 60.02GB to 59.52GB) and that on Server 2 is 0.50GB (from 58.88GB to 58.37GB), for 1 SIVM in each server. Even though the memory assigned to SIVM is 1.7GB, the actual memory consumed on the VM is around 0.19GB. This concludes that the free-memory reduction on Server 1 and Server 2 is the actual memory used by the VM and not the memory assigned to the VM.

The free-memory reduction on Server 3 is approximately 20.58GB (from 51.52GB to 30.94GB) for 2 SIVMs. The Ubuntu 10.04 LTS image file (with SSH access) created is approximately 11GB. This concludes that the free memory reduction on Server 3 is from the image file.

The free-memory reduction on Server 1 (0.50GB) and Server 2 (0.50GB) is small when compared to Server 3 (20.58GB) (from previous two paragraphs).

From my belief, the first linear decrease at 26s in free memory on Server 3 is due to the root disk image being cloned along with its directory and the file structure, and the cloned disk image is copied to the NFS. The second linear decrease in free memory at 134s on Server 3 occurs due to the creation of the swap space. Further investigation is needed for the final constant memory phase.

5.8.2 Comparing Free Memory for Adding Virtual Machine in the No-load and the Load Setups

Operation

For adding 2 SIVMs, one SIVM is added to Server 1 while the other SIVM is added to Server 2 in both setups.

Observation

Figure 16 shows the free memory for adding 2 SIVMs in the no-load and load setup. The duration for adding 2 SIVMs is 142 seconds and 154 seconds in the no-load and load setup, respectively.

In the no-load setup, the free memory on Server 3 is constant, 51.52GB (from 6s to 24s), and then there occurs a linear decrease in free memory from 51.52GB at 24s to 39.09GB at 36s, followed by another linear decrease in free memory from 39.09GB at 36s to 30.94GB at 135s, and once again the free memory becomes constant, 30.94GB (from 135s to 148s). In the load setup, the free memory on Server 3 is constant, 44.42GB (from 6s to 24s), and then there occurs a linear decrease in free memory from 44.42GB at 24s to 31.86GB at 32s, followed by another linear decrease in free memory from 31.86GB at 32s to 23.83GB at 142s, and once again the free memory becomes constant, 23.83GB (from 142s to 160s).
In the no-load setup, the free memory on Server 1 is constant, 60.02GB (from 6s to 146s), and then there occurs a linear decrease from 60.02GB at 146s to 59.52GB at 148s. In the load setup, the free memory on Server 1 is constant, 49.91GB (from 6s to 159s), and then there occurs a linear decrease from 49.91GB at 159s to 49.46GB at 163s.

In the no-load setup, the free memory on Server 2 is constant, 58.88GB (from 6s to 145s), and then there occurs a linear decrease from 58.88GB at 145s to 58.37GB at 148s. In the load setup, the free memory on Server 2 is constant, 37.63GB (from 6s to 154s), and then there occurs a linear decrease from 37.11GB at 154s to 37.11GB at 160s.

**Discussion and Conclusion**

The free-memory reduction on Server 1 is 0.50GB (from 60.02GB to 59.52GB) in the no-load setup and 0.45GB (from 49.91GB to 49.46GB) in the load setup. The free-memory reduction on Server 2 is 0.50GB (from 58.88GB to 58.37GB) in the no-load setup and 0.52GB (from 37.63GB to 37.11GB) in the load setup. This concludes that the free-memory reduction on Server 1 and Server 2 is approximately the same for both setups.

The free-memory reduction on Server 3 is approximately 21.58GB (from 51.52GB to 30.94GB) and 20.61GB (from 44.44GB to 23.83GB) in the no-load setup and the load setup, respectively. This concludes that the free-memory reduction on Server 3 does not change in the load and no-load setup.

The duration for the addition of 2 SIVMs in the load setup is 1.08 times more than the duration for the addition of 2 SIVMs in the no-load setup. This concludes that the duration linearly increases as the load in the machine increases.
5.8.3 Comparing Free Memory for Adding Different Virtual Machine Types in the No-load Setup

Operation
For adding 2 SIVMs, one SIVM is added to Server 1 while the other SIVM is added to Server 2. For adding 2 LIVMs, one LIVM is added to Server 1 while the other LIVM is added to Server 2.

Observation
Figure 17 shows the free memory for adding 2 SIVMs and 2 LIVMs in the no-load setup. The duration for adding 2 SIVMs and 2 LIVMs is 142 seconds and 168 seconds, respectively.

For 2 SIVMs, the free memory on Server 3 is constant, 51.52GB (from 6s to 24s), and then there occurs a linear decrease in free memory from 51.52GB at 24s to 39.09GB at 36s, followed by another linear decrease in free memory from 39.09GB at 36s to 30.94GB at 135s, and once again the free memory becomes constant 30.94GB (from 135s to 148s). For 2 LIVMs, the free memory on Server 3 is constant, 51.52GB (from 6s to 25s), and then there occurs a linear decrease in free memory from 51.52GB at 25s to 38.83GB at 36s, followed by another linear decrease in free memory from 38.83GB at 36s to 31.1GB at 160s, and once again the free memory becomes constant, 30.94GB (from 160s to 174s).

Figure 17: Free memory for adding 2 SIVMs and 2 LIVMs in the no-load setup
For 2 SIVMs, the free memory on Server 1 is constant, 60.02GB (from 6s to 146s), and then there occurs a linear decrease from 60.02GB at 146s to 59.52GB at 148s. For 2 LIVMs, the free memory on Server 1 is constant 55.70GB (from 6s to 169s), and then there occurs a linear decrease from 55.70GB at 169s to 54.85GB at 176s.
For 2 SIVMs, the free memory on Server 2 is constant, 58.88GB (from 6s to 145s), and then there occurs a linear decrease from 58.88GB at 145s to 58.37GB at 148s. For 2 LIVMs, the free memory on Server 2 is constant, 54.40GB (from 6s to 174s), and then there occurs a linear decrease from 54.40GB at 174s to 53.66GB at 179s.

**Discussion and Conclusion**

The free-memory reduction on Server 1 is 0.50GB (from 60.02GB to 59.52GB) for 1 SIVM and 0.85GB (from 55.70GB to 54.85GB) for 1 LIVM. The free-memory reduction on Server 2 is 0.50GB (from 58.88GB to 58.37GB) for 1 SIVM and 0.74GB (from 54.40GB to 53.66GB) for 1 LIVM. This concludes that the free-memory reduction on Server 1 and Server 2 is higher for the LIVM when compared to SIVM. Further investigation is needed to identify the cause of this increase, either from the change in RAM size or change in the number of CPU cores. I believe that this increase is from the change in RAM size and not from the change in the number of CPU cores.

The free-memory reduction on Server 3 is approximately 20.58GB (from 51.52GB to 30.94GB) for 2 SIVMs and 20.58GB (from 51.52GB to 30.94GB) for 2 LIVMs. This concludes that the memory reduction on Server 3 does not change as the VM type changes.

The duration for the addition of 2 LIVMs is 1.18 times more than the duration for the addition of 2 SIVMs. This concludes that the duration linearly increases for LIVMs when compared to SIVMs. Further investigation is needed to identify the cause of this increase, either from change in RAM size or change in the number of CPU cores. I believe that this increase is from the RAM size as images needs to be padded for the correct size.

**5.8.4 Comparing Free Memory for Adding Different Numbers of Virtual Machines in the No-load Setup**

**Operation**

For adding 2 SIVMs, one SIVM is added to Server 1 while the other SIVM is added to Server 2. For adding 8 SIVMs, four SIVMs are added to Server 1 while the other four SIVMs are added to Server 2.

**Observation**

Figure 18 shows the free memory for adding 2 SIVMs and 8 SIVMs in the no-load setup. The duration for adding 2 SIVMs and 8 SIVMs is 142 seconds and 897 seconds, respectively.

For 2 SIVMs, the free memory on Server 3 is constant, 51.52GB (from 6s to 24s), and then there occurs a linear decrease in free memory from 51.52GB at 24s to 39.09GB at 36s, followed by another linear decrease in the free memory from 39.09GB at 36s to 30.94GB at 135s, and once again the free memory becomes constant, 30.94GB (from 135s to 148s). For 8 SIVMs, the free memory on Server 3 is constant, 51.51GB (from 6s to 10s), and then there occurs a linear
decrease in free memory from 51.51GB at 10s to 38GB at 27s, followed by another linear decrease in free memory from 38GB at 27s to 10.33GB at 367s, and once again the free memory becomes constant, 10.33GB (from 367s to 440s). A recurrence of the pattern is observed at 441s for 8 SIVMs.

For 2 SIVMs, free memory on Server 1 reduces from 60.02GB to 59.52GB. For 8 SIVMs, free memory on Server 1 reduces from 59.64GB to 58.33GB. Four small dips in free memory are observed at 360s, 460s, 685s, and 910s for 8 SIVMs.

For 2 SIVMs, free memory on Server 2 reduces from 58.88GB to 58.37GB. For 8 SIVMs, free memory on Server 2 reduces from 58.49GB to 57.47GB. Four small dips in free memory are observed at 285s, 410s, 635s, and 860s for 8 SIVMs.

**Discussion and Conclusion**

The free-memory reduction on Server 1 is 0.50GB (from 60.02GB to 59.52GB) for 1 SIVMs and 1.31GB (from 59.64GB to 58.33GB) for 4 SIVMs, and on Server 2 is 0.50GB (from 58.88GB to 58.37GB) for 1 SIVM and 1.02GB (from 58.49GB to 57.47GB) for 4 SIVMs. This concludes that the free-memory reduction on Server 1 and Server 2 linearly increases as the number of VMs increases.

The free-memory reduction on Server 3 is approximately 21.04GB (from 51.52GB to 30.94GB) for 2 SIVMs and 51.40GB (from 51.51GB to 0.11GB) for 8 SIVMs. This concludes that the free-memory reduction on Server 3 linearly increases as the number of VMs increases.
The duration for adding 8 SIVMs is 6.31 times more than the duration for adding 2 SIVMs. This concludes that the duration linearly increases as the number of VMs increases. The reason for this increase is that OpenNebula is single threaded software; it will complete the first request before handling the second request.

For 8 SIVMs, four small dips in free memory on Server 1 and Server 2 correspond to 4 SIVMs each, as each SIVM uses the memory (RAM) to run the operating system. Further investigation is needed as the pattern (see observation 2nd paragraph) on Server 3 should appear 4 times instead of 2 times. Since OpenNebula is single threaded, the pattern appeared for 2 SIVMs on Server 3 should appear 4 times for 8 SIVMs. Refer discussion and conclusion from the section 5.8.1, for information about the activities performed during the pattern.

**5.8.5 Analyzing Redundant Memory for 2 Large-Instance Virtual Machines and 8 Small-Instance Virtual Machines in the No-load setup**

Redundant memory is the amount of memory used by the VM and this memory cannot be used for an application. The application is all software (e.g., web server, database server, load balancer, etc) apart from the operating system.

**Operation**

For adding 2 LIVMs, one LIVM is added to Server 1 while the other LIVM is added to Server 2. For adding 8 SIVMs, four SIVMs are added to Server 1 while the other four SIVMs are added to Server 2.

![Figure 19: Free Memory for Adding 2 LIVMs and 8 SIVMs in the no-load setup](image)
Observation

Figure 19 shows the free memory for Adding 2 LIVMs and 8 SIVMs in the no-load setup. The duration for the addition of 2 LIVMs and 8 SIVMs is 168 seconds and 897 seconds, respectively.

The free memory on Server 3 reduces from 51.52GB to 30.94GB and from 51.51GB to 0.11GB for 2 LIVMs and 8 SIVMs, respectively. The free memory on Server 1 reduces from 55.70GB to 54.85GB and from 59.64GB to 58.33GB for 2 LIVMs and 8 SIVMs, respectively. The free memory on Server 2 reduces from 54.40GB to 53.66GB and from 58.49GB to 57.47GB for 2 LIVMs and 8 SIVMs, respectively.

Discussion and Conclusion

The free-memory reduction on Server 1 is 0.85GB (from 54.67GB to 55.09GB) for 1 LIVM and 1.31GB (from 57.71GB to 58.07GB) for 4 SIVMs, on Server 2 is 0.87GB (from 53.79GB to 53.37GB) for 1 LIVM and 1.02GB (from 56.56GB to 56.92GB) for 4 SIVMs. This concludes that the free-memory reduction per virtual machine on Server 1 and Server 2 for 1 LIVM is higher (2.65 times) when compared to 4 SIVMs. This infers that the redundant memory for LIVM is higher when compared to multiple SIVMs.

The duration for adding 8 SIVMs is longer (5.33 times) when compared to 2 LIVMs. The duration linearly increases as the number of VMs increases. However it is up to the user to consider the tradeoff between the redundant memory and the duration.

5.8.6 Evaluating Free Memory for Deleting Virtual Machines in the No-load Setup

Operation

For deleting 2 SIVMs, one SIVM is deleted from Server 1 while the other SIVM is deleted from Server 2.

Observation

Figure 20 shows the free memory for deleting 2 SIVMs in the no-load setup. The duration for deleting 2 SIVMs is 3 seconds.

The free memory on Server 3 shows a linear increase from 30.47GB to 51.51GB. The free memory on Server 1 increases from 59.42GB to 59.63GB, while the free memory on Server 2 increases from 58.27GB to 58.48GB.

Discussion and Conclusion

The free-memory increase on Server 1 is 0.21GB (from 59.42GB to 59.63GB) and that on Server 2 is 0.21GB (from 58.27GB to 58.48GB), for 1 SIVM in each server. Even though the memory assigned to SIVM is 1.7GB, the actual memory consumption on the VM is around 0.19GB. This
concludes that the free-memory increase on Server 1 and Server 2 is from the actual memory used by the VM and not the memory assigned to the VM.

![Figure 20: Free memory for deleting 2 SIVMs in the no-load setup](image)

The free-memory increase on Server 3 is approximately 21.04GB (from 30.47GB to 51.51GB) for 2 SIVMs. The Ubuntu 10.04 LTS image file (with SSH access) created is approximately 11GB. This concludes that the free-memory increase on Server 3 is from the image file.

The free-memory increase on Server 1 (0.21GB) and Server 2 (0.21GB) is small when compared to Server 3(21.04GB) (from previous two paragraphs).

5.8.7 Comparing Free Memory for Deleting Virtual Machine in the No-load and the Load Setups

Operation

For deleting 2 SIVMs, one SIVM is deleted from Server 1 while the other SIVM is deleted from Server 2 in both setups.
Figure 21: Free memory for deleting 2 SIVMs in the no-load and load setup

**Observation**

Figure 21 shows the free memory for deleting 2 SIVMs in the no-load setup and load setup. The duration for deleting 2 SIVMs in the load setup and no-load setup is 3 seconds.

The free memory on Server 3 linearly increases from 30.47GB to 51.51GB and from 23.36GB to 44.41GB in the no-load and load setup, respectively.

The free memory on Server 1 increases from 59.42GB to 59.63GB and from 49.30GB to 49.52GB in the no-load and load setup, respectively. The free memory on Server 2 increases from 58.27GB to 58.48GB and from 36.96GB to 37.20GB in the no-load and load setup, respectively.

**Discussion and Conclusion**

The free-memory increase on Server 1 is 0.21GB (from 59.42GB to 59.63GB) in the no-load setup and 0.21GB (from 49.30GB to 49.52GB) in the load setup. The free-memory increase on Server 2 is 0.21GB (from 58.27GB to 58.48GB) in the no-load setup and 0.24GB (from 36.96GB to 37.20GB) in the load setup.
to 37.20GB) in the load setup. This concludes that the free-memory increase on Server 1 and Server 2 is approximately the same for both setups.

The free-memory increase on Server 3 is approximately 21.04GB (from 30.47GB to 51.51GB) and 21.04GB (from 23.36GB to 44.41GB) in the no-load and load setup, respectively. This concludes that the free-memory increase on Server 3 does not change in the no-load and load setup.

The duration for deleting 2 SIVMs is 3 seconds for both setups. This concludes that the duration for the delete operation is not affected by the load, as the application’s state will be deleted ungracefully.

5.8.8 Comparing Free Memory for Deleting Different Virtual Machine Types in the No-load Setup

Operation
For deleting 2 SIVMs, one SIVM is deleted from Server 1 while the other SIVM is deleted from Server 2. For deleting 2 LIVMs, one LIVM is deleted from Server 1 while the other LIVM is deleted from Server 2.

Observation
Figure 22 shows the free memory for deleting 2 SIVMs and 2 LIVMs in the no-load setup. The duration for deleting 2 SIVMs and 2 LIVMs is 3 seconds, respectively.

The free memory on Server 3 increases linearly, from 30.47GB to 51.51GB and from 30.03GB to 51.50GB for 2 SIVMs and 2 LIVMs, respectively.

The free memory on Server 1 increases from 59.42GB to 59.63GB and from 54.67GB to 55.09GB for 2 SIVMs and 2 LIVMs, respectively. Free memory on Server 2 increases from 58.27GB to 58.48GB and from 53.37GB to 53.37GB for 2 SIVMs and 2 LIVMs, respectively.

Discussion and Conclusion
The free-memory increase on Server 1 is 0.21GB (from 59.42GB to 59.63GB) for 1 SIVM and 0.42GB (from 54.67GB to 55.09GB) for 1 LIVM, on Server 2 is 0.21GB (from 58.27GB to 58.48GB) for 1 SIVM and 0.42GB (from 53.37GB to 53.37GB) for 1 LIVM. This concludes that the free-memory increase on Server 1 and Server 2 is higher for LIVM when compared for SIVM. Further investigation is needed to identify the cause of this increase, either from the change in RAM size or change in the number of CPU cores. I believe that this increase is from the RAM size change and not from the change in the number of CPU cores.
The free-memory increase on Server 3 is approximately 21.04GB (from 30.47GB to 51.51GB) for 2 SIVMs and 21.47GB (from 30.03GB to 51.50GB) for 2 LIVMs. This concludes that the free memory increase on Server 3 does not change with VM types.

![Free memory for deleting 2 SIVMs and 2 LIVMs in the no-load setup](image)

Figure 22: Free memory for deleting 2 SIVMs and 2 LIVMs in the no-load setup

The duration for deleting 2 SIVMs and 2 LIVMs is 3 seconds. This concludes that the duration for delete operation is not affected by VM types, as the application’s state will be deleted ungracefully.

### 5.8.9 Comparing Free Memory for Deleting Different Numbers of Virtual Machines in the No-load Setup

**Operation**

For deleting 2 SIVMs, one SIVM is deleted from Server 1 while the other SIVM is deleted from Server 2. For deleting 8 SIVMs, four SIVMs are deleted from Server 1 while the other four SIVMs are deleted from Server 2.
Observation

Figure 23 shows the free memory for deleting 2 SIVMs and 8 SIVMs in the no-load setup. The duration for deleting 2 SIVMs and 8 SIVMs is 3 seconds and 7 seconds, respectively.

Figure 23: Free memory for deleting 2 SIVMs and 8 SIVMs in the no-load setup

The free memory on Server 3 linearly increases from 30.47GB to 51.51GB and from 0.24GB to 51.62GB for 2 SIVMs and 8 SIVMs, respectively.

The free memory on Server 1 increases from 59.42GB to 59.63GB and from 57.71GB to 58.07GB for 2 SIVMs and 8 SIVMs, respectively. The free memory on Server 2 increases from 58.27GB to 58.48GB and from 56.56GB to 56.92GB for 2 SIVMs and 8 SIVMs, respectively.

Discussion and Conclusion

The free-memory increase on Server 1 is 0.21GB (from 59.42GB to 59.63GB) for 1 SIVM and 0.36GB (from 57.71GB to 58.07GB) for 4 SIVMs, on Server 2 is 0.21GB (from 58.27GB to 58.48GB) for 1 SIVM and 0.36GB (from 56.56GB to 56.92GB) for 4 SIVMs. This concludes that the free-memory increase on Server 1 and Server 2 is linear as the number of VMs increases.
The free-memory increase on Server 3 is approximately 21.04GB (from 30.47GB to 51.51GB) for 2 SIVMs and 51.38GB (from 0.24GB to 51.62GB) for 8 SIVMs. This concludes that the free-memory increase on Server 3 is linear as the number of VMs increases.

The duration for deleting 8 SIVMs is longer (2.3 times) when compared to 2 SIVMs. This concludes that the duration linearly increases as the number of VMs increases. The reason behind this increase is that OpenNebula is single threaded; it will complete the first request before handling the second request.

5.8.10 Analyzing Redundant Memory for 2 Large-Instance Virtual Machines and 8 Small-Instance Virtual Machines in the No-load setup

Redundant memory is the amount of memory used by the VM and this memory cannot be used for an application. The application is all software (e.g., web server, database server, load balancer, etc) apart from the operating system.

**Operation**

For deleting 2 LIVMs, one LIVM is deleted from Server 1 while the other LIVM is deleted from Server 2. For deleting 8 SIVMs, four SIVMs are deleted from Server 1 while the other four SIVMs are deleted from Server 2.

**Observation**

Figure 24 shows the free memory for deleting 2 LIVMs and 8 SIVMs in the no-load setup. The duration for deleting 2 LIVMs is 3 seconds and 8 SIVMs is 7 seconds.

The free memory on Server 3 increases linearly from 30.03GB to 51.50GB and from 0.24GB to 51.62GB for 2 LIVMs and 8 SIVMs, respectively.

The free memory on Server 1 increases from 54.67GB to 55.09GB and from 57.71GB to 58.07GB for 2 LIVMs and 8 SIVMs, respectively. The free memory on Server 2 increases from 53.79GB to 53.37GB and from 56.56GB to 56.92GB for 2 LIVMs and 8 SIVMs, respectively.

**Discussion and Conclusion**

The free-memory increase on Server 1 is 0.42GB (from 54.67GB to 55.09GB) for 1 LIVM and 0.36GB (from 57.71GB to 58.07GB) for 4 SIVMs, on Server 2 is 0.42GB (from 53.79GB to 53.37GB) for 1 LIVM and 0.36GB (from 56.56GB to 56.92GB) for 4 SIVMs. This concludes that the free-memory increase per virtual machine on Server 1 and Server 2 for 1 LIVM is greater (4.1 times) when compared to 4 SIVMs. This infers that the redundant memory for LIVM is higher when compared to multiple SIVMs.

The duration for deleting 8 SIVMs is longer (2.3 times) when compared to 2 LIVMs. The duration linearly increases as the number of VMs increases. However it is up to the user to consider the tradeoff between the redundant memory and the duration.
Figure 24: Free memory for deleting 2 LIVMs in no-load setup and 8 SIVMs in the no-load setup

5.8.11 Evaluating Network Transfer for the Live Migration of Virtual Machines in No-load Setup

Operation

Eight SIVMs are migrated from Server 2 to Server 1.

Observation

Figure 25 shows the network transfer per second for live migration of 8 SIVMs from Server 2 to Server 1. The duration for live migration of 8 SIVMs is 122 seconds. The total amount of data transferred from Server 2 is equal to the total amount of data received in Server 1. As a result, the data received on Server 1 is not shown in the graph. There is no significant change in the network transfer of Server 3.
We can observe the following pattern and this pattern reoccurs 8 times at 6s, 26s, 44s, 56s, 71s, 86s, 103s, and 116s. From 0MB/s, the network transfer per second increases to 12MB/s and drops back to 0MB/s. I refer to this as the first cycle. From 0MB/s, the network transfer per second increases linearly to 36MB/s and drops back linearly to 0MB/s. I refer to this as the second cycle.

![Network transfer per second for the live migration of 8 SIVMs in the no-load setup from Server 2 to Server 1](chart.png)

Figure 25: Network transfer per second for the live migration of 8 SIVMs in the no-load setup from Server 2 to Server 1

**Discussion and Conclusion**

From observation 2\(^{nd}\) paragraph, the network traffic is not high in the first cycle (transferred 0.04GB in 12s, adding network transmit from 6s to 18s), but it is higher in the second cycle (transferred 0.19GB in 7s, adding network transmit from 19s to 26s). We can infer that the duration of the pattern is not constant. This affects the prediction of the total migration time.

From the observation, we can infer that the network transfer of Server 3 is unaffected during the migration.
Live migration is a movement of VM data from the source server (Server 2) to the destination server (Server 1). Memory consumption in each SIVM is approximately 0.19GB. Therefore, for 8 SIVMs the memory consumption is approximately 1.52GB (0.19GB*8). The amount of data transferred for 8 SIVMs in the live migration is 1.93GB (adding network transmit from 6s to 128s). The total amount of page dirtied for 8 SIVMs is approximately 0.41GB (1.93GB - 1.52GB). This verifies that the live migration includes the transfer of the dirty pages to the destination (as discussed in table 16 for live migration).

5.8.12 Comparing Network Transfer for the Live Migration of Virtual Machines in the No-load and Load Setups

Operation

Eight LIVMs are migrated from Server 2 to Server 1 in both setups.

![Live migration of 8 LIVMs in no-load setup](image)

![Live migration of 8 LIVMs in load setup](image)

Figure 26: Network transfer per second for the live migration of 8 LIVMs from Server 2 to Server 1 in the no-load (a) and load setup (b).

Observation

Figure 26 shows the network transfer per second for the live migration of 8 LIVMs from Server 2 to Server 1 in the no-load (a) and load setup (b). The duration for the live migration of 8 LIVMs is 242 seconds and 238 seconds in the no-load and load setup, respectively. The maximum value of the network transfer for 8 LIVMs is 33MB/s and 50MB/s in the no-load and load setup, respectively.

We can observe the following pattern, for the no-load setup (figure 26.a) and this pattern reoccurs 8 times. From 0MB/s, the network transfer per second linearly increases to 33MB/s and then it linearly decreases to 2MB/s. From 2MB/s, the network transfer per second linearly
increases to 9MB/s and then it linearly decreases to 2MB/s. From 2MB/s the network transfer per second linearly increases to 33MB/s and then it linearly decreases to 0MB/s.

An inverse parabolic curve can be observed in figure 26.b. From 20MB/s, the network transfer per second linearly increases to 50MB/s and drops back linearly to 20MB/s.

**Discussion and Conclusion**

The maximum value of the network transfer for 8 LIVMs in the load setup (50MB/s) is higher when compared to the no-load setup (33MB/s). This concludes that the maximum value of the network transfer increases linearly as the load in the machine increases.

The increase (from 0MB/s to 20MB/s) in the network transfer in the load setup is from the Olio load (figure 26.b).

The duration for the live migration of 8 LIVMs in the load and the no-load setup is approximately constant. This concludes that the duration for the live migration is not affected by the load.

For the load setup, no pattern is observed and the network transfer varies based on the Olio load. This makes it difficult for an administrator to choose the specification of the network equipment (such as, 1 Mbps or 1 Gbps switch).

### 5.8.13 Comparing Network Transfer for the Live Migration of Different Virtual Machine Types in the No-load Setup

**Operation**

Eight SIVMs and eight LIVMs are migrated separately from Server 2 to Server 1 in the no-load setups.

**Observation**

Figure 27 shows the network transfer per second for the live migration of 8 SIVMs (a) and 8 LIVMs (b) from Server 2 to Server 1. The duration for the live migration of 8 SIVMs is 122 seconds and the duration for the live migration of 8 LIVMs is 242 seconds.

We can observe the following pattern and this pattern reoccurs 8 times. For 8 SIVMs (figure 27.a) from 0MB/s, the network transfer per second increases linearly to 12 MB/s and then it drops back linearly to 0MB/s. From 0MB/s, the network transfer per second increases linearly to 36MB/s and drops back linearly to 0MB/s. For 8 LIVMs (figure 27.b), from 0MB/s, the network transfer per second linearly increases to 33MB/s and then it linearly decreases to 3MB/s. From 3MB/s, the network transfer per second linearly increases to 9MB/s and then it linearly decreases to 3MB/s. From 3MB/s, the network transfer per second linearly increases to 33MB/s and then it linearly decreases to 0MB/s.
Discussion and Conclusion

The amount of data transferred for 8 SIVMs for the live migration is 1.93GB (adding network transmit from 6s to 128s, figure 27.a). The amount of data transferred for 8 LIVMs for the live migration is 3.72GB (adding network transmit from 6s to 248s, figure 27.b). This concludes that the amount of data transfer for LIVM increases (twice) when compared to SIVM. Further investigation is needed to identify the cause of this increase, either from the change in RAM size or change in the number of CPU cores. I believe that this increase is from the RAM size change and not from the change in the number of CPU cores.

The maximum value of the network transfer for 8 SIVMs (36MB/s) is higher when compared to 8 LIVMs (33MB/s). This concludes that the maximum value of the network transfer for SIVM linearly increases when compared to LIVM. Further investigation is needed to identify the cause of this increase, either from the change in RAM size or change in the number of CPU cores. I believe that this increase is from the RAM size.

The duration for the live migration of 8 LIVMs is longer (1.98 times) when compared to 8 SIVMs in the no-load setup. This concludes that the duration for the live migration linearly increases for LIVM when compared to SIVM. Further investigation is needed to identify the cause of this increase, either from the change in RAM size or change in the number of CPU cores. I believe that this increase is from the RAM size.
5.8.14 Evaluating Network Transfer for the Cold Migration of Virtual Machine in the No-load Setup

Operation

Eight LIVMs are cold migrated from Server 1 to Server 2.

![Network transfer per second for the cold migration of 8 LIVMs in the no-load setup from Server 1 to Server 2](image)

Figure 28: Network transfer per second for the cold migration of 8 LIVMs in the no-load setup from Server 1 to Server 2

Observation

Figure 28 shows the network transfer per second for the cold migration of 8 LIVMs in the no-load setup from Server 1 to Server 2. The duration for the cold migration of 8 LIVMs is 227 seconds. There is no network traffic in certain intervals (such as from 6s to 30s).

The total amount of data transferred from Server 1 is equal to the total amount of data received in Server 3. As a result, the data received in Server 3 is not shown in the graph. We can observe a pattern and this pattern reoccurs 8 times at 30s, 58s, 84s, 114s, 140s, 170s, 198s, and 224s. From
0MB/s, the network transfer per second linearly increases to 113 MB/s and drops back linearly to 0MB/s. I refer to this as the first cycle.

The total amount of data transferred from Server 3 is equal to the total amount of data received in Server 2. As a result, the data received in Server 2 is not shown in the graph. We can observe a pattern and this pattern reoccurs 8 times at 32s, 61s, 86s, 117s, 143s, 173s, 200s, and 226s. From 3MB/s, the network transfer per second linearly increases to 117MB/s and then it linearly decreases to 3MB/s. From 3MB/s, the network transfer per second linearly increases to 10MB/s and then it linearly decreases to 3MB/s. From 3MB/s, the network transfer per second linearly increases to 117MB/s and then it linearly decreases to 0MB/s. I refer to this as the second cycle.

**Discussion and Conclusion**

The network traffic is higher in the first cycle (transferred 0.46GB in 5s, adding network transmit from 31s to 35s), but it is lower in the second cycle (transferred 0.48GB in 17s, adding network transmit from 31s to 48s).

We can observe no network traffic in certain intervals (e.g., from 6s to 30s). This verifies that the VM is stopped before migration (as discussed in table 16 for cold migration).

From the observation in the 2\textsuperscript{nd} and 3\textsuperscript{rd} paragraph, we can infer that the duration of the pattern is not constant. For example, from the observation 2\textsuperscript{nd} paragraph, the first pattern starts at 30s and second pattern starts at 58s. The difference is 28s (58s – 30s). So the third pattern should start at 86s (58s + 28s), but from the observation the third pattern starts at 84s. This affects the prediction of the total migration time.

Memory consumption in each LIVM is approximately 0.28GB (measured before running experiment using Linux command), therefore, for 8 LIVMs, memory consumption is approximately 2.24GB (0.28*8). The amount of data transferred in Server 1 for 8 LIVMs for the cold migration is 3.7GB (adding network transmit of Server 1 from 6s to 233s). The state file that is transferred for 8 LIVMs is approximately 1.46GB (3.7GB – 2.24GB). This verifies that the cold migration includes the transfer of the state file to the destination (as discussed in table 16 for cold migration).

The cold migration is the movement of VM data from the source server (Server 1) to Server 3 and from Server 3 to the destination server (Server 2). The amount of data transferred for 8 LIVMs from Server 1 to Server 3 is 3.7GB (adding network transmit of Server 1 from 6s to 233s) and from Server 3 to Server 2 is 3.5GB (adding network transmit of Server 3 from 6s to 233s). The total amount of data transferred for 8 LIVMs for the cold migration is 7.2GB. This concludes that the cold migration uses twice the network transfer (network transfer from Server 1 to Server 3 + network transfer from Server 3 to Server 2).
5.8.15 Comparing Cold Migration and Live Migration of Virtual Machine in the No-load Setup

Operation

Eight LIVMs are migrated from Server 1 to Server 2 during the cold migration and the eight LIVMs are migrated from Server 2 to Server 1 during the live migration in the no-load setup.

![Cold migration of 8 LIVMs in no-load setup](image)

![Live migration of 8 LIVMs in no-load setup](image)

Figure 29: Network transfer per second for the cold migration (a) and the live migration (b) of 8 LIVMs in the no-load setup

Observation

Figure 29 shows the network transfer per second for the cold migration (a) and the live migration (b) of 8 LIVMs. The duration for the cold migration of 8 LIVMs is 227 seconds and for the live migration of 8 LIVMs is 242 seconds.

For the cold migration of 8 LIVMs from Server 1 to Server 2 (figure 29.a), the total amount of data transferred from Server 1 is equal to the total amount of data received in Server 3. As a result, the data received in Server 3 is not shown in the graph. We can observe a pattern and this pattern reoccurs 8 times. From 0MB/s, the network transfer per second linearly increases to 113 MB/s and drops back linearly to 0MB/s. The total amount of data transferred from Server 3 is equal to the total amount of data received in Server 2. As a result, the data received in Server 2 is not shown in the graph. We can observe a pattern and this pattern reoccurs 8 times. From 3MB/s, the network transfer per second linearly increases to 117MB/s and then it linearly decreases to 3MB/s. From 3MB/s, the network transfer per second linearly increases to 10MB/s and then it linearly decreases to 3MB/s. From 3MB/s, the network transfer per second linearly increases to
117MB/s and then it linearly decreases to 0MB/s. We can observe no network traffic in certain intervals (e.g., from 6s to 30s).

For the live migration of 8 LIVMs from Server 2 to Server 1 (figure 29.b), the total amount of data transferred from Server 2 is equal to the total amount of data received in Server 1. As a result, the data received in Server 1 is not shown in the graph. We can observe the following pattern and this pattern reoccurs 8 times. From 0MB/s, the network transfer per second linearly increases to 33MB/s and then it linearly decreases to 0MB/s. From 0MB/s, the network transfer per second linearly increases to 9MB/s and then it linearly decreases to 0MB/s. From 0MB/s, the network transfer per second linearly increases to 33MB/s and then it linearly decreases to 0MB/s.

**Discussion and Conclusion**

The amount of data transferred for 8 LIVMs for the cold migration is 7.2GB (adding network transmit from 6s to 233s, figure 29.a). The amount of data transferred for 8 LIVMs for the live migration is 3.72GB (adding network transmit from 6s to 248s, figure 29.b). This concludes that the amount of data transfer in the cold migration is higher (twice) when compared to the live migration.

We can observe no network traffic in certain intervals (e.g., from 6s to 30s) for the cold migration when compared to the live migration. I believe that, during this interval the VM is stopped and prepared for the cold migration and this is not seen in the live migration.

The maximum value of the network transfer for the cold migration of 8 LIVMs is 117MB/s and for the live migration of 8 LIVMs is 33MB/s. This concludes that the maximum value of the network transfer in the cold migration is higher (3.5 times) when compared to the live migration.

The duration for the live migration of 8 LIVMs is longer (1.06 times) when compared to the cold migration of 8 LIVMs. This concludes that the cold migration is faster when compared to the live migration.

**5.8.16 Comparing Network Transfer for the Cold Migration of Virtual Machine in the No-load Setup and Load Setup**

**Operation**

Eight SIVMs are cold migrated from Server 2 to Server 1 in both setups.

**Observation**

Figure 30 shows the network transfer per second for the cold migration of 8 SIVMs in the no-load (a) and load setup (b) from Server 1 to Server 2. Duration for the cold migration of 8 SIVMs is 103 seconds and 110 seconds in the no-load and load setup, respectively. There is no network traffic from 6s to 16s in both setups and from 20s to 27s in the load setup.
Figure 30: Network transfer per second for the cold migration of 8 SIVMs in the no-load (a) and load setup (b) from Server 1 to Server 2

The total amount of data transferred from Server 1 is equal to the total amount of data received in Server 3. As a result, the data received in Server 3 is not shown in the graph. For the no-load setup (figure 30.a), from 0MB/s, the network transfer per second linearly increases to 118MB/s and drops back linearly to 0MB/s. For the load setup (figure 30.b), from 0MB/s, the network transfer per second linearly increases to 103MB/s and drops back linearly to 0MB/s. We can see this pattern reoccurs 8 times in both setups.

The total amount of data transferred from Server 3 is equal to the total amount of data received in Server 2. As a result, the data received in Server 2 is not shown in the graph. For the no-load setup (figure 30.a), from 0MB/s, the network transfer per second linearly increases to 46MB/s and then it linearly decreases to 7MB/s. From 7MB/s, the network transfer per second linearly increases to 117MB/s and then it linearly decreases to 0MB/s. For the load setup (figure 30.b), from 0MB/s, the network transfer per second linearly increases to 41MB/s and then it linearly decreases to 9MB/s. From 9MB/s, the network transfer per second linearly increases to 103MB/s and then it linearly decreases to 0MB/s. We can see this pattern reoccurs 8 times in both setups.

**Discussion and Conclusion**

The maximum value of the network transfer for 8 SIVMs in the no-load setup (117MB/s) is higher when compared to the load setup (103MB/s). This concludes that the maximum value of the network transfer linearly decreases as the load in the machine increases.

In the cold migration of virtual machine from Server 2 to Server 1, after Server 3 receives virtual machine’s data and state file from Server 2, Server 3 will immediately send virtual machine’s data and state file to Server1. Further investigation is needed as there is no network traffic from
20s to 27s in the load setup, but in this time frame Server 3 should have sent the virtual machine’s data to Server 1 but from the measurement it is sent after 27s.

The duration for the cold migration of 8 SIVMs in the load setup is longer (1.06 times) when compared to the no-load setup. This concludes that the duration for the cold migration linearly increases as the load in the machine increases.

5.8.17 Comparing Network Transfer for the Cold Migration of Different Virtual Machine Types in the No-load Setup

Operation

Eight SIVMs and eight LIVMs are cold migrated separately from Server 1 to Server 2 in the no-load setups.

![Cold migration of 8 SIVMs in no-load setup](image1)

![Cold migration of 8 LIVMs in no-load setup](image2)

Figure 31: Network transfer per second for the cold migration of 8 SIVMs (a) and 8 LIVMs (b) from Server 1 to Server 2 in the no-load setup

Observation

Figure 31 shows the network transfer per second for the cold migration of 8 SIVMs (a) and 8 LIVMs (b) from Server 1 to Server 2. The duration for the cold migration of 8 SIVMs is 103 seconds and the duration for the cold migration of 8 LIVMs is 227 seconds.

The total amount of data transferred from Server 1 is equal to the total amount of data received in Server 3. As a result, the data received in Server 3 is not shown in the graph. For 8 SIVMs (figure 31.a), from 0MB/s, the network transfer per second linearly increases to 118MB/s and drops back linearly to 0MB/s. For 8 LIVMs (figure 31.b), from 0MB/s, the network transfer per
second linearly increases to 113 MB/s and drops back linearly to 0MB/s. We can see this pattern reoccurs 8 times in both setups.

The total amount of data transferred from Server 3 is equal to the total amount of data received in Server 2. As a result, the data received in Server 2 is not shown in the graph. For 8 SIVMs (figure 31.a), from 0MB/s, the network transfer per second linearly increases to 46MB/s and then it linearly decreases to 7MB/s. From 7MB/s, the network transfer per second linearly increases to 117MB/s and then it linearly decreases to 0MB/s. For 8 LIVMs (figure 31.b), from 3MB/s, the network transfer per second linearly increases to 117MB/s and then it linearly decreases to 0MB/s. From 3MB/s, the network transfer per second linearly increases to 10MB/s and then it linearly decreases to 3MB/s. From 3MB/s, the network transfer per second linearly increases to 117MB/s and then it linearly decreases to 0MB/s. We can see this pattern reoccurs 8 times in both setups.

**Discussion and Conclusion**

The amount of data transferred for 8 SIVMs for the cold migration is 3.87GB (adding network transmit from 6s to 109s, figure 31.a). The amount of data transferred for 8 LIVMs for the cold migration is 7.2GB (adding network transmit from 6s to 233s, figure 31.b). This concludes that the amount of data transfer increases (twice) for LIVM when compared to SIVM. Further investigation is needed to identify the cause of this increase, either from the change in RAM size or change in the number of CPU cores. I believe that this increase is from the RAM size change and not from the change in the number of CPU cores.

The maximum value of the network transfer for 8 SIVMs and 8 LIVMs is approximately constant. This concludes that the maximum value of the network transfer does not change even if the virtual machine type changes for the cold migration.

The duration for the cold migration of 8 LIVMs is longer (2.20 times) when compared to 8 SIVMs. This concludes that the duration for the cold migration linearly increases for LIVM when compared to SIVM. Further investigation is needed to identify the cause of this increase, either from the change in RAM size or change in the number of CPU cores. I believe that this increase is from the RAM size.

**5.9 Overall Conclusion**

The overall comparison for different virtual machine management operations is summarized in table 17, 18, 19 and 20.

**5.9.1 Major Contribution**

The major lessons learned related to the performance evaluation are: (1) the duration for the live migration does not change with the load; (2) the duration for the live migration increases linearly
as the memory assigned to the VM increases; (3) the duration of the add and delete operations increases linearly as the number of VMs increases.

Table 17: Summary of observations for adding virtual machine (VM)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Scenario</th>
<th>Conclusion</th>
</tr>
</thead>
</table>
| Add                      | Basic operation (scenario 5.8.1)              | 1. The memory that Cloud Slave consumes is the actual memory used by the VM and not the memory assigned to the VM.  
2. The memory that Cloud Master consumes has the same size as the image file. |
| Comparing between different VM types (scenario 5.8.3) | 1. The memory that Cloud Slave consumes linearly increases as the memory assigned to VM increases.  
2. The memory that Cloud Master consumes does not change with the VM types.  
3. The duration linearly increases as the memory assigned to the VM increases. |
| Comparing between no-load setup and load setup (scenario 5.8.2) | 1. The memory that Cloud Slave consumes does not change with the load.  
2. The memory that Cloud Master consumes does not change with the load.  
3. The duration linearly increases as the load on Cloud Slave increases. |
| Comparing between different number of VMs (scenario 5.8.4) | 1. The memory that Cloud Slave consumes linearly increases as the number of VMs increases.  
2. The memory that Cloud Master consumes linearly increases as the number of VMs increases.  
3. The duration linearly increases as the number of VMs increases. |
Table 18: Summary of observations for deleting virtual machine (VM)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Scenario</th>
<th>Conclusion</th>
</tr>
</thead>
</table>
| Delete          | Basic operation (scenario 5.8.6)| 1. The memory released on Cloud Slave is the actual memory used by the VM and not the memory assigned to the VM.  
                              |                                  | 2. The memory released on Cloud Master has the same size as the image file. |
|                 | Comparing between different VM types (scenario 5.8.8) | 1. The memory released on Cloud Slave linearly increases as the memory assigned to the VM increases.  
                              |                                  | 2. The memory released on Cloud Master does not change with the VM type.  
                              |                                  | 3. No change in duration, as the application’s state will be deleted ungracefully. |
|                 | Comparing between no-load setup and load setup (scenario 5.8.7) | 1. The memory released on Cloud Slave does not change with the load.  
                              |                                  | 2. The memory released on Cloud Master does not change with the load.  
                              |                                  | 3. No change in duration, as the application’s state will be deleted ungracefully. |
|                 | Comparing between different number of VMs (scenario 5.8.9) | 1. The memory released on Cloud Slave linearly increases as the number of VMs increases.  
                              |                                  | 2. The memory released on Cloud Master linearly increases as the number of VMs increases  
                              |                                  | 3. The duration increases linearly as the number of VMs increases. |
Table 19: Summary of observations for live migration of virtual machine (VM)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Scenario</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Live Migrate</td>
<td>Basic operation (scenario 5.8.11)</td>
<td>1. The Cloud Master is involved only in triggering the VM live migration operation and not involved in actual transfer of the VM&lt;br&gt;2. Live migration of a VM on Cloud Slave involves two different kinds of network traffic. When comparing the two kinds, one has higher amount of data transfer with shorter duration when compared to another.</td>
</tr>
<tr>
<td></td>
<td>Comparing between no-load setup and load setup (scenario 5.8.12)</td>
<td>1. The maximum transfer rate linearly increases as the load on Cloud Slave increases.&lt;br&gt;2. The duration for the live migration does not change with the load.</td>
</tr>
<tr>
<td></td>
<td>Comparing between different VM types (scenario 5.8.13)</td>
<td>1. The amount of data transfer linearly increases as the memory assigned to the VM increases.&lt;br&gt;2. The maximum transfer rate does not change with the VM types.&lt;br&gt;3. The duration for the live migration linearly increases as the memory assigned to the VM increases.</td>
</tr>
</tbody>
</table>
Table 20: Summary of observations for cold migration of virtual machine (VM)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Scenario</th>
<th>Conclusion</th>
</tr>
</thead>
</table>
| Cold Migrate      | Basic operation (scenario 5.8.14)             | 1. The Cloud Master is involved in triggering the VM cold migration operation and in transferring the VM.  
2. Cloud Master receives a VM from the Cloud Slave has a higher rate of data transfer with shorter duration (from Cloud Slave X to Cloud Master) when compared to transfer of the VM to another Cloud Slave (from Cloud Master to Cloud Slave Y). |
|                   | Comparing between no-load setup and load setup (scenario 5.8.16) | 1. The maximum transfer rate linearly decreases as the load on Cloud Slave increases.  
2. The duration of the cold migration linearly increases as the load on Cloud Slave increases. |
|                   | Comparing between different VM types (scenario 5.8.17) | 1. The amount of data transfer linearly increases as the memory assigned to the VM increases.  
2. The maximum transfer rate does not change with the VM types.  
3. The duration for the cold migration linearly increases as the memory assigned to the VM increases. |
| Live Migrate & Cold Migrate | Comparing between live migration and cold migration (scenario 5.8.15) | 1. The amount of data transfer in the cold migration is twice (from Cloud Slave X to Cloud Master + from Cloud Master to Cloud Slave Y) when compared to the live migration (directly from Cloud Slave X to Cloud Slave Y).  
2. The maximum transfer rate in the cold migration is higher when compared to the live migration.  
3. The duration for the live migration is longer when compared to the cold migration. |
6 Conclusions and Future Work

6.1 Conclusion

This thesis’s focus has been on two major goals. The first goal is to develop a framework for comparing the features of the private-IaaS-cloud management software. The second goal is to evaluate the performance of the OpenNebula private-IaaS-cloud management software.

The first goal is achieved by developing a framework for comparing the features of the private-IaaS-cloud management software. The major contribution related to the framework is identifying features provided for the commercial software that are not available for the open-source software. The features are: (1) co-location of VMs is running a group of VMs on the same physical machine (for example, if the web server VM has to access the application server VM for getting the web pages, they can be placed on the same physical machine); (2) anti-co-location of VMs is not allowing a pair of VMs to run on a single physical machine (for example, the primary and back-up web server VMs should always run on different physical machines); (3) the resources of the physical machines can be combined (e.g., number of CPU cores, physical memory) as a resource pool and compartmentalized into an organizational structure (e.g., HR, development, testing, etc).

The second goal is achieved by evaluating the performance of the OpenNebula private-IaaS-cloud management software. The major lessons learned related to the performance evaluation are: (1) the duration for the live migration does not change with the load; (2) the duration for the live migration increases linearly as the memory assigned to the VM increases; (3) the duration of the add and delete operations increases linearly as the number of VMs increases.

6.2 Experiences

The framework part of the thesis project was planned to be developed with ten cloud management software packages (five open-source and five commercial software packages). Instead of taking more features and software into consideration, a specific feature could have been evaluated in more detail. For example, contextualization could be evaluated for all possible options. It would be ideal to take three software and eight features for comparison.

The performance evaluation was planned to be conducted with five cloud management software packages (three open-source and two commercial software packages). Since many of the cloud management software do not provide Java API, the cloud operation code needs to be written in different programming languages (e.g., C, Python). Since I am not familiar with other programming languages, I decided to evaluate the OpenNebula software in detail with different testing scenarios.

The Cloud Operation Code is written in Java using the OpenNebula Java API (see section 5.4). This Code is specific to OpenNebula and cannot be used with other cloud management software. Instead of using a software specific API, the code can be written using a generic API (e.g., OCCI.
The generic API is supported by many cloud management software packages. For more information about the supported API, see section 4.3.5.

In this thesis, a virtual-machine type has two parameters (number of CPU cores and RAM size). When comparing the free-memory usage for different virtual-machine types by having one parameter with a fixed value (fixed number of CPU cores with different RAM size), will reduce the uncertainty in the free-memory usage as discussed in this thesis (see section 5.8.8). The free-memory usage could be either from the change in RAM size or from the change in the number of CPU cores.

### 6.3 Future Work

The most popular hypervisors are Xen and KVM. In this thesis the performance of OpenNebula software is evaluated based on the KVM hypervisor. Evaluating the performance of OpenNebula software with Xen as the hypervisor and comparing that result with the result provided by this thesis will enable comparison of the performance of OpenNebula software with different hypervisors.

The performance evaluation can be conducted with different cloud management software (OpenStack, Eucalyptus, Nimbus, etc.). This extended performance evaluation allows the cloud consumer to choose the software based on performance metrics, such as the minimum duration for creating a VM.
References


[64] OpenStack software. Accounting. [Online]. Available: 
http://docs.openstack.org/cactus/openstack-compute/starter/content/Project_Management-d1e2672.html. Last access on 15/2/2012.

http://docs.openstack.org/cactus/openstack-compute/admin/content/components-of-openstack.html. Last access on 15/2/2012.

http://docs.openstack.org/cactus/openstack-compute/starter/content/Functions_and_Features-d1e132.html. Last access on 15/2/2012.

[67] OpenStack software. Scheduling types. [Online]. Available: 
http://docs.openstack.org/cactus/openstack-compute/starter/content/Scheduler_nova-scheduler-_d1e262.html. Last access on 15/2/2012.

http://wiki.abiquo.com/display/user/%27Best+Fit%27+resource+Scheduler. Last access on 15/2/2012.

http://community.abiquo.com/display/ABI17/Installing+abicli+-+Abiquo+Command+Line+Interface. Last access on 15/2/2012.

http://community.abiquo.com/display/ABI17/15+Minute+Tutorial. Last access on 15/2/2012.

http://community.abiquo.com/display/ABI17/Installing+abicli+-+Abiquo+Command+Line+Interface. Last access on 15/2/2012.

http://community.abiquo.com/display/ABI17/Manage+Enterprises#ManageEnterprises-ResourceAllocationLimits. Last access on 15/2/2012.

http://community.abiquo.com/display/ABI17/Manage+Users. Last access on 15/2/2012.

[74] Apache Olio PHP. About Apache Olio PHP install. [Online]. Available: 

http://wiki.abiquo.com/display/ABI17/Abiquo+Documentation+Home. Last access on 15/2/2012.


Appendix A

The commands used for the performance evaluation of OpenNebula cloud management software are listed here. [84]

**Virtual Machine Management**

Add: To create a virtual machine (VM)
   `VirtualMachine.allocate(Client client, String description)`
   client - XML-RPC client and description - template of the virtual machine, this returns

Migrate: To cold or live migrate a VM
   `VirtualMachine vm = new VirtualMachine(int ID, Client client)`
   `vm.migrate(int HID, boolean T/F)`
   client - XML-RPC client, ID - the virtual machine id, HID – the host ID where VM will run and T/F – true for live migration and false for cold migration.

Delete: To remove a VM
   `VirtualMachine vm = new VirtualMachine(int ID, Client client)`
   `vm.finalizeVM()`
   client - XML-RPC client and ID - the virtual machine id