NTNU – Trondheim
Norwegian University of
Science and Technology

Evaluation of a Private Cloud for Higher Education

Trygve Andre Tønnesland

Master of Science in Informatics
Submission date: June 2013
Supervisor: Trond Aalberg, IDI
Co-supervisor: Anders Christensen, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

In modern day research and education there is a constant need for dedicated server installations, both for permanent and for temporary use by the research staff and by students. Typically there is need for access to computational power, network connectivity and specialized software.

The classical solution for this problem is to delegate physical hardware for all of these needs. This is unpractical and results in an inefficient use of hardware, electricity and administrative resources.

A self-service virtualization system could benefit these institutions by making computing resources more easily available to its students and researchers, and by improving the utilization of hardware resources.

In this project we identify the requirements for such a system, by installing and configuring a prototype private cloud solution for students at the Department of Computer and Information Science. The solution we've chosen is based on OpenStack, an open source cloud operating system. The implementation is evaluated through monitoring over a four month period and through surveys. We suggest an approach for implementing a permanent solution, and highlight challenges that need to be considered.

Our experimentation has shown that OpenStack is a suitable system for implementing a private cloud solution, even though the project is still under heavy development. Through our surveys and user feedback we can determine that a user demand for virtualization service exists.

Sammendrag

I dagens forskning- og undervisningshverdag er det et konstant behov for dedikerte tjenerinstallasjoner, både for permanent og midlertidig bruk av forskere og av studenter. Behovet består av tilgang til beregningskraft, nettverkstilkobling og spesialisert programvare.

Den klassiske løsningen på dette problemet er å dele ut fysisk maskinvare for å dekke disse behovene. Dette er upraktisk og fører til en ineffektiv bruk av maskinvare, elektrisitet og administrative ressurser.

En selvbetjent virtualiseringsløsning kan være nyttig for disse institusjonene, ved å gjøre maskinressurser lettere tilgjengelig for studenter og forskere, og ved å forbedre utnyttelsen av maskinvare.

I dette prosjektet identifiserer vi kravene til en slik løsning, samt installerer og konfigurerer en prototype av en privat skyløsning for studenter og ansatte ved Institutt for datateknikk og informasjonsvitenskap. Vår løsning er basert på OpenStack, et skysystem med åpen kildekode. Implementasjonen er evaluert gjennom spørreundersøkelser og gjennom overvåkning gjennom en fire måneder lang periode. Vi anbefaler en fremgangsmåte for en permanent løsning, og bemerker utfordringer som må vurderes.

Vårt eksperiment har vist at OpenStack er et passende system for implementasjon av en privat skytjeneste, selv om prosjektet fremdeles er under kraftig utvikling. Gjennom våre spørreundersøkelser og tilbakemeldinger fra brukere kan vi fastslå at det finnes et behov for en slik virtualiseringstjeneste.

Foreword

This Master's thesis is the final part of a Master of Science degree from the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU).

I would like to thank my supervisors Trond Aalberg and Anders Christensen for their support and guidance, and thanks for giving me the opportunity to shape my own project. I would also like to thank Arne Dag Fidjestøl at the technical group for giving me helpful advice, and for providing necessary equipment to perform my experiments.

Finally I would like to express my appreciation to my fellow students for trying the solution, which has given me necessary data and feedback to perform my analysis.

June 1, 2013
Trygve André Tønnesland

Contents

List of Figures

List of Tables

Acronyms

AWS Amazon Web Services. 7, 9, 22

EBS Elastic Block Storage. 7, 23

EC2 Elastic Cloud Computing. 7

IaaS Infrastructure as a Service. 9, 10, 21, 22, 64

KVM Kernel-based Virtual Machine. 30, 36, 37

LDAP Lightweight Directory Access Protocol. 24, 35

LVM Logical Volume Manager. 30

NFS Network File System. 30

PaaS Platform as a Service. 8

S3 Simple Storage Service. 7

SaaS Software as a Service. 8

SAN Storage Area Network. 30, 45

Chapter 1

Introduction

During the last few years open source virtualization systems have evolved quickly. Many of these projects are sponsored by large companies, contributing with both financial support and developers. This results in an active development and as these systems become more mature, it is interesting to see how they will perform in a real-world installation.

1.1 Problem description

In modern day research and education there is a constant need for dedicated server installations, both for permanent and for temporary use by the research staff and by students. Typical needs are access to computational power, network connectivity and specialized software.

The classical solution for this problem is to delegate physical hardware for all of these needs. This is unpractical and results in inefficient use of hardware, electricity and administrative resources.

A self-service virtualization system could benefit these institutions by making computing resources more easily available to its students and researchers, and by improving the utilization of hardware resources.

1.2 Project goal

The goal of this project is to evaluate and implement a prototype private cloud solution based on open source software with regards to the needs of the staff and students at the Department of Computer and Information Science (IDI), at Norwegian University of Technology and Science (NTNU).

The focus of this project is the situation at IDI, but our methods and results are applicable for other higher education institutions.

We will suggest an approach for implementing a permanent private cloud solution in an educational context, in addition to methods for evaluating the success of such systems.

The objectives of our analysis are:

1. What characterizes the requirements for a virtualization platform with regards to an educational institution and its users?
2. How can you evaluate the success of a private cloud solution deployed in an educational context?
3. What are the challenges that need to be considered when deploying a widely available virtualization platform?
4. Can a virtualization platform improve the availability of computing resources to students and researchers?
5. How can a virtualization platform improve the utilization of available resources?

1.3 Approach

The approach for this project is to consider requirements and use cases for an open source virtualization system in an educational context, to install and configure an OpenStack environment, and to maintain and monitor a prototype installation with a number of users over a period of time. The goal of this approach is to get necessary feedback from users and experience with the solution. The final goal is to provide recommendations for future work and a permanent deployment.

1.4 Report outline

This chapter describes the task, and defines the scope of the project. Chapter 2 dwells into the background of virtualization history and cloud computing, and also presents related work on this topic. Chapter 3 describes the requirements to the system to be implemented, and we present the technology implemented in our project. The implementation and configuration of the system are described in Chapter 4, and Chapter 5 presents the operational period and our experience from operating the system. Chapter 6 presents the surveys conducted while preparing for the operation period, and later while evaluating the project. Conclusions and future work are presented in Chapter 7.

Chapter 2

Virtualization and cloud computing

In this chapter we will introduce the concept of virtualization and cloud computing, and go through the history of virtualization. We will also present related work on use of cloud computing in higher education.

2.1 Virtualization

Different computer resources can be virtualized. When using the term *virtualization* in this report, we refer to *hardware virtualization*. Other examples of virtualization are network virtualization with virtual LAN segments (VLAN) and VPN, and application virtualization where applications run on a remote server and only the screen image is transferred over the network to the client.

Hardware virtualization (or platform virtualization) is the virtualization of computers or operating systems. This is performed by software components, a *hypervisor* or *virtual machine monitor*, creating a virtual machine running in a confined environment. This can be achieved using different approaches:

Full virtualization. With full virtualization the virtual machine simulates the full hardware of a physical machine, allowing an unmodified guest operating system to

be run without modification. This is done by utilizing binary translation [1] to rewrite non-virtualizable instructions.

Paravirtualization. Paravirtualization is a technique where instead of translating non-virtualizable instructions, the hypervisor provides an interface to the virtual machine allowing execution of hypercalls replacing privileged instructions [2]. This will in many cases give a performance benefit compared to translation of instructions, but the drawback is that it requires modification of the guest operating system.

Hardware assisted virtualization. Hardware assisted virtualization utilize specialized CPU extensions to reduce the amount instructions that needs to be translated. This introduces a significant performance increase compared to traditional full virtualization [3].

Operating system-level virtualization. Operating system level-virtualization utilizes a kernel allowing multiple isolated instances of the user-land, instead of just one. These instances are often referred to as *containers* or *jails*. This removes the overhead of emulating a complete hardware platform. The limitation of this method is that the virtual machines are restricted to running the same kernel as the host computer.

2.1.1 History

The concept of virtualization has its roots from the 1960s, with the introduction of the mainframe system IBM 7044, also known as *"the M44"*. The M44 became the basis for IBM M44/44X, an experimental computer system. The system was a purely research system, simulating multiple M44 virtual machines. This system gave each user a running virtual machine with an image of the 44X operating system. The purpose of this was among others to evaluate the concepts of virtualization and time sharing systems [4].

The idea was to allow the resources to be time-shared between multiple users, so that the mainframe computers could be used more efficiently. This was accomplished by suspending programs that were waiting for peripheral resources, and allowing other programs that utilize different resources to run in the meantime.

Virtualization was for a long time a concept only available with mainframe solutions, partly because of the huge amount of resources in the mainframes, but also because most server applications were developed for mainframes, and therefore no real reason to pursue this technology outside the mainframe sphere.

The mainframes were big and expensive, and often required custom operating sys-

tems and specially compiled programs. This, combined with the increasing use of computing in everyday business activities resulted in a demand for smaller and less expensive alternative to the old mainframes.

As the demand for less expensive server got higher, server manufacturers started to produce servers based on the Intel x86 architecture. This was an architecture already used for desktop computers, and a familiar technology to the developer community. When the need for computing resources increased in the 1990s, the x86 architecture became the most popular one because of its lower cost. The combined power of multiple low-cost servers and distributed client-server applications provided performance comparable to the old mainframe systems. With the shift towards x86 based servers, the idea of virtualization was nearly abandoned.

Gordon Moore once stated that the number of transistors on integrated circuits doubles every year. This was later revised to every two years [5] and is commonly known as Moore's law. Moore's law is now cited as "computing performance doubling every 18 months", a quote made famous by Intel employee Dave House [6]. This prediction has turned out to be true for the most part of the history of modern computing.

The ever increasing computing performance leaves us with a large amount of unused resources, as hardware development tends to surpass software development. The situation with resource utilization is the same as the early mainframes, where most of the resources are left unused for most of the time.

The first efforts for x86 virtualization was driven by the need for running software developed for different operating systems concurrently on the same machine. In 1999, Kevin P. Lawton published a paper about x86 virtualization [7]. In the paper Lawton outlined three different strategies for achieving this: Pure emulation, OS/API emulation and virtualization. In his paper he discussed some of the challenges related to virtualizing the x86 architecture, which include but are not limited to incompatible instructions. The same year VMware released their first product, the VMware Virtual Platform, making it possible to run multiple operating systems on a single physical desktop machine.

Modern day x86-server virtualization started in 2001 when VMware released their first server virtualization products. The main goal for this, was the same as the old mainframe systems, namely to improve the utilization of the available hardware resources.

Seeing VMware's success with their products and the high demand for such solutions, processor manufactures started to develop virtualization support built into the CPUs. In 2005 Intel released their first CPUs with dedicated virtualization extensions (VT-x), and in 2006 AMD followed up with the release of AMD-V. The goal of these hard-

ware extensions was to reduce the number of instructions to be emulated in software, allowing a more lightweight hypervisor and increasing the performance of the virtualized machine. In 2008 the second generation of hardware supported virtualization was unveiled with Intel Extended Page Tables (EPT) and AMD Nested Page Tables (NPT), a technology reducing the overhead with virtualization by offloading the memory management from the hypervisor to the CPU [8]. Other improvements include technologies as Intel Virtualization for Connectivity (VT-c), which integrates virtualization support in the network adapter and improves data throughput by offloading this task from the CPU [9].

With the added hardware support for virtualization, the technology has developed quickly the last decade. Further increase in computing power per CPU socket, decreased memory price and overall improved point to point network connectivity, have made it more feasible to move traditionally locally hosted services to the "Cloud".

2.2 Cloud computing

Cloud computing is a relatively new term used to describe shared hardware and software resources delivered to the end-users as a service over a network.

The cloud technology emerged from major internet companies like Amazon who wanted to capitalize on renting out capacity in their infrastructure. Amazon launched their cloud services in 2006 when announcing Simple Storage Service (S3) providing a high availability and scalable storage system accessible through web services. The same year Elastic Cloud Computing (EC2) launched, allowing users to rent virtual machines. Later more products were added, e.g. Elastic Block Storage (EBS) providing persistent storage to machines running in EC2. These products are all parts of Amazon Web Services (AWS)[1], a stack of services available as a public cloud. The common denominator for these services is that they all can be controlled through the AWS API. Through Amazons success with AWS, the AWS API has become the de-facto standard for cloud solutions.

The most commonly used definition of the term cloud computing is the one provided by National Institute of Standards and Technology (NIST) [10] which is quoted in the following:

> Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing re-

[1] `http://aws.amazon.com`

> sources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

The definition also denotes the following characteristics of a cloud computing system:

On-demand self-service. A consumer may adjust the provisioned resources without requiring human interaction with the service provider.

Broad network access. The provided resources are accessed over the network using standard mechanisms.

Resource pooling. The computing resources (e.g. processing, memory, storage, and network bandwidth) of the provider are shared between numerous consumers, and are dynamically assigned and reassign according to consumer demand.

Rapid elasticity. The amount of resources provided to a consumer can rapidly be scaled (in or out), often automatically, according to consumer demand. To the consumer, the available resources may appear to be unlimited and can be allocated at any time.

Measured service. The utilization of computing resources can be monitored, controlled and reported, allowing the provider to charge their consumers on a per-use basis.

2.2.1 Service models

The NIST definition [10] divides cloud computing into three different service models. The service models characterizes the different levels a cloud platform can operate, from providing an application to a complete infrastructure where the consumer maintains most parts of the stack, from the network configuration to the hosted application.

Software as a Service (SaaS). The capability provided is an application running on a cloud infrastructure. The consumer typically accesses this application trough a web browser, and does not manage or control the underlying infrastructure. Examples: Google Drive and Office 365.

Platform as a Service (PaaS). The capability provided to the consumer is the ability to deploy self-made or acquired applications on a cloud infrastructure. The application must be created using programming languages, libraries and tools supported by the

provider. The consumer controls the deployed application, but not any part of the underlying infrastructure. Examples: Cloud Foundry, Google App Engine, Heroku and Microsoft Azure.

Infrastructure as a Service (IaaS). The capability provided is the ability to provision computing resources (e.g. processing, memory, storage and networking) used to deploy and run arbitrary applications. The consumer controls the operating system, storage, deployed applications, and often selected network components (e.g. host firewalls and private networks). Examples: Amazon EC2 and Rackspace Cloud Servers.

2.2.2 Deployment models

Clouds can be deployed internally in an organization, or be provided as a service by a third party. NIST [10] divides the deployment models into four different combinations of public and private solutions.

Private cloud. A cloud infrastructure utilized by a single organization. The infrastructure may be managed by the organization itself, or by a third-party provider. Examples: Eucalyptus, OpenStack and VMware vCloud.

Community cloud. A cloud infrastructure shared by several organizations in a community with shared interests and concerns.

Public cloud. A cloud infrastructure available to the general public through a third party service provider. Examples: AWS, Google App Engine, Heroku, Rackspace Cloud Servers, etc.

Hybrid cloud. A cloud infrastructure which consists of two or more cloud infrastructures (private, community, or public). The infrastructures are connected by standardized or proprietary technology enabling data and application portability allowing load balancing or migration between clouds.

2.3 Related work

In the recent years there have been many publications on different applications of virtualization and cloud computing, both in general and in research and educational contexts.

One research direction is about analyzing the performance of virtual computers. In the paper *"VM consolidation: A real case based on OpenStack cloud"* Corradi et al. research the performance degradation when applying load to multiple virtual computers running on the same physical host [11]. Another research direction with regards to performance is benchmarking different private IaaS solutions, like in the paper *"Cloud computing performance benchmarking and virtual machine launch time"* where Steinmetz et al. analyze the provisioning time of OpenStack and Eucalyptus [12]. Similar research has also been done by von Laszwski et al. described in the paper *"Comparison of Multiple Cloud Frameworks"*. In their study they do scalability experiments on different IaaS frameworks in the FutureGrid[2] infrastructure. They find that the different solutions fit different requirements.

A research direction closer to the scope of this thesis is the different applications of private cloud technology in academia. During the last years there have been many publications on this topic.

In the paper *"Practical Cloud Evaluation from a Nordic eScience User Perspective"* Åke Edlund describes the findings of the NEON[3] project [13]. The project experimented in offloading non-HPC jobs (low memory and few CPU cores) from the traditional HPC systems to both public and private clouds. In their findings it shows that about 20% of the jobs currently running could be suitable for such offloading.

In Germany, Hochschule Furtwangen University (HFU) have deployed a private Cloud Infrastructure called *CloudIA*, allowing employees and students to provision virtual computers, and on demand collaboration software [14]. Their experience shows that such solutions can improve the management of computers used for programming exercises.

In a project closer to home, a group of students at Gjøvik University College implemented an OpenStack solution for use as a framework for virtual computer labs [15]. Their focus was to customize the web interface for batch operations. In their conclusion they find OpenStack suitable for implementing a private cloud, but emphasize that it is a complicated system, requiring thoroughly planning before implementation.

Earlier there have been two separate specialization projects at IDI looking into how the use of virtualization could benefit the department.

In 2008 Frode Sandholtbråten did a study on virtualization of terminal rooms [16]. In his project he did a study on replacing the traditional terminal rooms with personal

[2] `https://portal.futuregrid.org/`
[3] `http://www.necloud.org/`

virtual machines. His proposal was to use VMware ESX(i) to provide this service.

In 2009 Andreas Eriksen wrote about and implemented a self-service virtualization system based on XenServer and a custom web interface to administer the virtual machines [17]. During his project, Andreas did a survey showing that there was a user demand for a virtualization service.

Chapter 3

Project planning

This chapter covers the planning of this project. We will define requirements for a private cloud solution for an educational institution. We will also go through the current situation regarding virtualization and personal servers at IDI, and also present previous projects on this topic. Figure 3.1 shows the different stages of the project.

We will also present the architecture and design of OpenStack and its different components, and alternative products.

The chosen platform for this project is OpenStack. OpenStack was chosen early on due to it being the open source system with the most active community while receiving heavy commercial backing.

Figure 3.1 – Project process

3.1 Current situation at IDI

As an educational and research institution, the Department of Computer and Information Science often differ from most other organizations when considering require-

ments to technical solutions.

The most significant differences from other organizations are:

- Large number of temporary users (students)
- High degree of diversity of the individual ongoing projects
- Small IT-organization compared to total number of users

Currently IDI at NTNU has about 800 students, divided on the five year integrated master's program in computer engineering, the three year bachelor's program in computer science and the two year master's programs. About 300 of these students are in their two last years of their education. A number of these students need a computer to host their projects and run their applications related to their work. This need is also applicable for the PhD candidates and researchers at the department. The common solution to this problem is either to use personal computers, or to get assigned a workstation or a personal server from the technical group.

As most of the students have a laptop as their primary computer, many of them need a personal workstation or a server to run applications or calculations that require certain running time.

The last year the technical group has handed out 30 physical workstations and about 20 virtual computers in the existing VMware installation to students at IDI. This covers about half of the current fifth year master students. There is no advertising for the current VMware service, so the user demand is likely to be higher.

The existing solution for virtualization provides no interface where the end user can provision their own virtual machines or control their machines remotely, other than what provided by the operating system running on the virtual machine. This requires involvement from an administrator for both provisioning of machines, and when access to the local console is needed.

Earlier projects at IDI have studied implementation of a self-service virtualization system for the department. In 2009 Andreas Eriksen implemented a solution based on XenServer and a custom web interface allowing users to provision and control their own virtual machines [17]. His study showed that there was a user demand for such a service.

Since 2009 more products for providing virtualization solutions have become available. This project follows up the previous work, by studying how such a solution can be implemented using modern technology.

3.2 Requirements

In this section we will specify the requirements most important for a virtualization system used in an educational environment. The requirements are designed in collaboration with the technical staff at IDI. In the process of identifying the requirements we have designed use cases describing the central tasks for the system.

The following definitions are used this context:

- System: Software components providing virtualization services
- Administrator: The person or IT-department responsible for maintaining the system
- User: Students, staff and faculty members with a need for server and application hosting.
- Virtual machine: A machine hosted in a confined environment at the system.

3.2.1 Use cases

This section provides a description of the most central tasks for a virtualization system using textual use cases.

In Figure 3.2 we show a diagram of the user and an administrator performing common tasks, including provisioning and controlling virtual machines, quota management and system maintenance.

Use Case 1	**Provisioning of virtual machines**
Actor:	End-User
Stakeholders:	• System administrator: The process should involve a minimum of administrative overhead
Preconditions:	The end-user must be logged into the system

Postconditions:	The virtual machine should either be installed or sent for approval by the system administrator

Main Success Scenario:

1. The end-user selects ”create new virtual machine” in the interface
2. Fills out required hardware specification and operating system
3. The virtual machine gets provisioned and login information gets delivered to the end-user

Extensions:

3.a Insufficient hardware quota
   1. System returns a message informing the user that the virtual machine has been sent for approval by the system administrator
   2. Process ends

3.b Invalid specifications
   1. System shows failure message
   2. User returns to step 2 and corrects the errors

Use Case 2	**Control provisioned machine**
Actor:	End-User
Stakeholders:	• System administrator: The process should involve a minimum of administrative overhead
Preconditions:	• The end-user must be logged into the system • The virtual machine must be provisioned
Postconditions:	

Main Success Scenario:

1. The end-user selects an existing virtual machine in the interface
2. The end-user choose to open a console
3. The end-user logs on to the virtual machine and perform his or hers tasks

Extensions:

3.a The virtual machine is not responding as expected
   1. Choose to reboot the virtual machine in the interface
   2. User returns to step 2
   3. If the problem persists, file a service call or reprovision the virtual machine

3.b Invalid credentials for the virtual machine
   1. File a service call to get a password reset, or reprovision the virtual machine
   2. User returns to step 2

Use Case 3	**Approve provision of virtual machines**
Actor:	System administrator
Stakeholders:	• End-User: Wants the virtual machine to be provisioned without unnecessary delay
Preconditions:	• The system administrator must be logged onto the system • An end-user must have submitted a request to provision a virtual machine
Postconditions:	The virtual machine should be provisioned

Main Success Scenario:

1. The system administrator approves the request to provision a virtual machine
2. The virtual machine gets provisioned, according to the specifications given by the end-user
3. The end-user gets notified that the virtual machine is ready for use

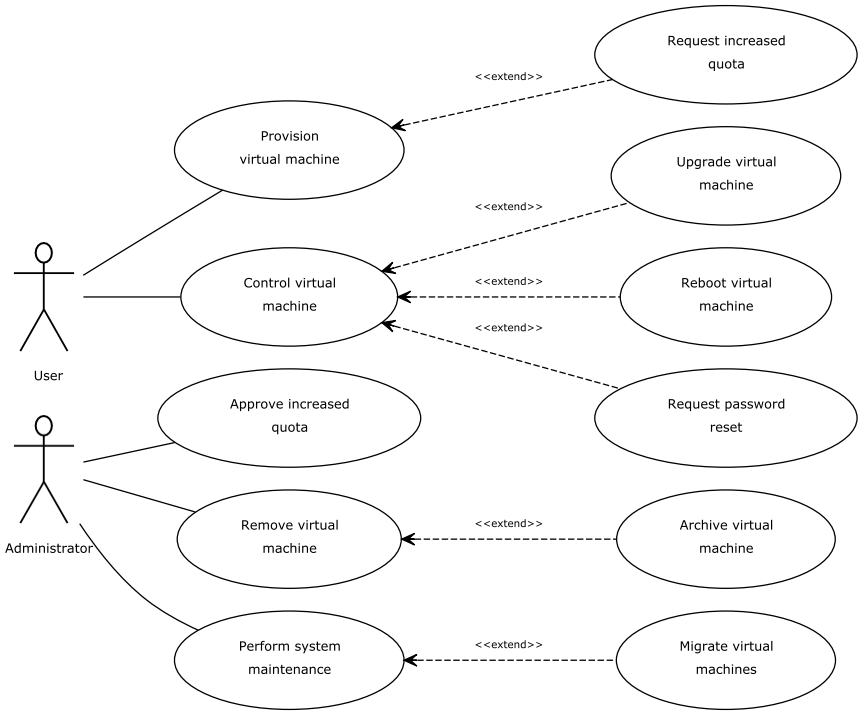

Figure 3.2 – Use case diagram of the system

Use Case 4	**Perform routine maintenance of the virtualization system**
Actor:	System administrator
Stakeholders:	• End-User: The maintenance should have a minimum of impact on the services provided to the users
Preconditions:	

Postconditions:	The system should be updated

Main Success Scenario:

1. The system administrator sets a node in the system in maintenance mode
2. The virtual machines hosted at that node gets migrated to other nodes
3. The system administrator performs maintenance at the node

Use Case 5	**Manual patching of virtual machines**
Actor:	End-user
Stakeholders:	• System administrator: All virtual machines hosted by the system should have the latest security patches installed.
Preconditions:	The end-user should be logged into the virtual machine with his or hers administrator account
Postconditions:	The virtual machine should be updated

Main Success Scenario:

1. The end-user choose to install all pending updates
2. The end-user restarts the virtual machine, if required

Use Case 6	**Removal of virtual machines**
Actor:	System administrator
Preconditions:	The system administrator should be logged onto the system

Postconditions: The virtual machine should be removed from the system, and archived if needed

Main Success Scenario:

1. The system administrator chooses to remove the virtual machine

Extensions:

1.a The virtual machine should be archived before removal
   1. Choose to archive the virtual hard drive
   2. Returns to step 1.

3.2.2 Primary requirements

These are the primary requirements to the system:

PRQ1 **Users must be able to provision and control their own virtual machines, with a minimum involvement from the administrator.** This is to ensure that the users get the resources they need when they need it, and to reduce the administrative overhead.

PRQ2 **The system must be free to use and open source.** There should not be any licensing costs related to the system, and it should be possible to modify the system if necessary.

PRQ3 **The system must be able to run on standard x86 hardware.** The system should not require any specialized hardware. This is to allow the department to re purpose existing hardware.

PRQ4 **If the system requires a separately installed operating system, the software components must be available through the distributions software repository.** This to ensure maintainability, and that security updates will be easily available.

PRQ5 **The system must have a level of maturity, with regards to stability, available documentation, user base and online community.** This to ensure that support is available.

PRQ6 **The system must be actively maintained by the developers.** This to ensure that the system will be provided with necessary updates for the foreseeable future.

PRQ7 **The system must have support for common guest operating system, including Linux, FreeBSD and Windows guests.** These are the primary operating systems maintained by the technical group at IDI.

3.2.3 Secondary requirements

These are the secondary requirements to the system:

SRQ1 **The system should either provide its own operating system (bare-bone) or run on an operating system already maintained by the technical department at the institute for computer science.** This to ensure that we don't introduce a system unfamiliar to the department.

SRQ2 **The system should use existing services for authenticating and authorizing its users.** This to avoid the administrative overhead of distributing login credentials to new users.

SRQ3 **The system should be designed to allow routine maintenance of its components with a minimum of down time on the virtual machines hosted at the system.** This is to reduce downtime, and improve the flexibility of performing maintenance.

SRQ4 **The system should have a web interface that allows provisioning and control of virtual machines.** A web interface will allow the users to interact with the system without installing custom software on their computers.

SRQ5 **The system should have routines, automatic or manual, to ensure that active virtual machines are maintained automatically or by the users. Virtual machines without proper security updates should be disabled.** This is to avoid that the virtual machines running on the system present a security risk for the organizations infrastructure.

SRQ6 **The system should have routines, automatic or manual, to ensure that virtual machines owned by students and staff without active affiliation to the organization are disabled in an orderly fashion.** This to avoid that the systems resources are being used by third parties.

SRQ7 **The system should have routines, automatic or manual, to ensure that disabled virtual machines can be preserved for future use.** The purpose of preserving machines is to allow others to follow up projects.

SRQ8 **Commercial support for the system should be available.** Commercial support can in many cases be a good supplement to community resources.

SRQ9 **The system should have the ability to restrict a virtual machine to a single IP address.** Address restriction is important to avoid IP-conflicts or hijacking due to misconfiguration or malicious use.

SRQ10 **The system should have the ability to generate reports showing provisioned virtual machines and their owners.** Reports showing the system usage can be an important tool for capacity planning.

SRQ11 **Users without a quota should be able to register the need for a virtual machine. This should generate a notification to the system administrators.** In cases where users need resources that extend from the assigned quota, the user should easily be able to request increased quota.

3.3 Available technology

There are a number of different open source virtualization projects in active development. Some of these projects have commercial backing, and are widely available in leading Linux distributions.

In the following sections we will present OpenStack and some of the alternative solutions for private IaaS clouds.

3.4 OpenStack

OpenStack[1] is a cloud computing project founded by Rackspace, NASA and others in June 2010 [18]. Currently the project is supported by major companies like AT&T, IBM, and Canonical [19], and the software components are available in both the Ubuntu and Red Hat Linux distributions.

The project initially started with combining code from the Nebula platform developed by NASA and Rackspace's Cloud Files Platform. The first official release was made four months after the project started. Since then there have been regular releases every few months, and new major releases twice a year. Figure 3.3 shows a timeline of major releases.

[1] `http://www.openstack.org`

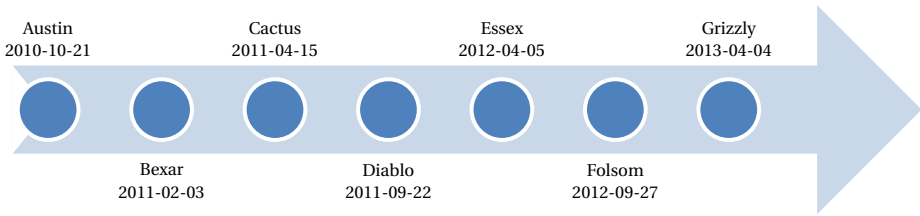

Figure 3.3 – Timeline of OpenStack releases [20]

In 2011 OpenStack took Eucalyptus' place as the default cloud service in the Ubuntu Enterprise Cloud-project [21], giving the project a head position among the open source cloud systems.

3.4.1 Components

OpenStack consists of several independent components written in Python, that each perform different roles in a virtualization environment. The goal is to provide a full set of services from management interfaces, through networking and storage, to computing. Most of the components in OpenStack are API-compliant with AWS, allowing users to choose between a large variety of tools for administering their cloud. Figure 3.4 shows the conceptual architecture of the Folsom release of OpenStack.

3.4.1.1 Compute (Nova)

Compute provides the main component of an IaaS system, namely the cloud computing fabric controller [22]. As a fabric controller Compute facilitates the hypervisor and manages the available resources. The Nova project originated out of NASA Ames Research Laboratory [23], and took its natural place in OpenStack when NASA and RackSpace joined forces in 2010. The service provided by Nova is comparable to Amazon's EC2.

Modern operating systems are designed for, and expect its own dedicated hardware. To be able to virtualize you need some sort of virtual hardware to run this on. This is done by the hypervisor, either with full hardware virtualization, or a middle, paravirtualization. What Compute does is to provide a flexible interface for controlling the virtualization hypervisor. An important idea behind Compute, is that it should be both hardware and hypervisor agnostic. Currently Nova supports KVM, LXC, QEMU, UML, VMware ESX(i), Xen, PowerVM, Hyper-V and bare metal servers [24].

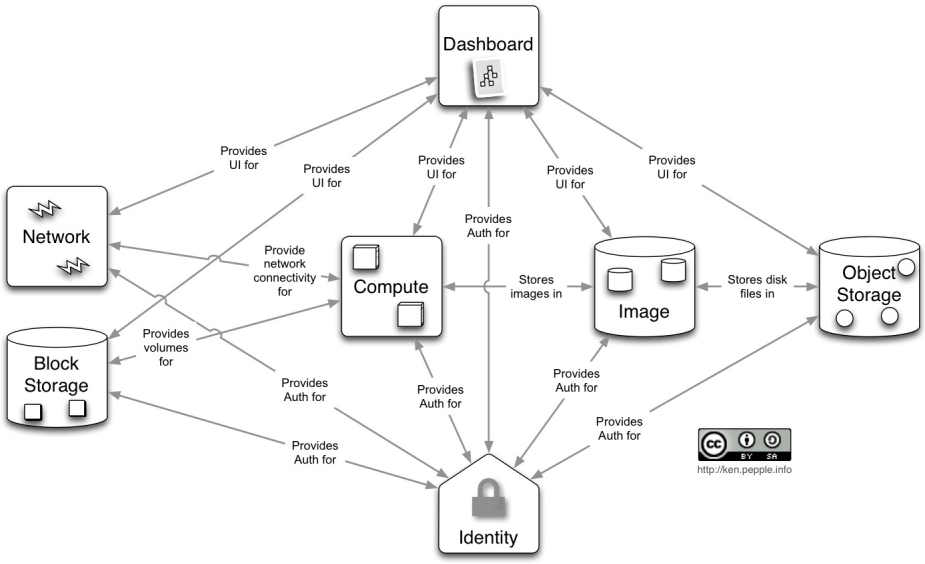

Figure 3.4 – OpenStack conceptual architecture. *Source:* `http://ken.pepple.info` *under CC-license*

In addition to the bare virtualization, Compute also provides basic networking configurations and volatile[2] block level storage for the virtual hard drives through image files.

3.4.1.2 Storage

OpenStack provides two different solutions for persistent storage, each for different type of applications.

Block Storage (Cinder). Cinder provides persistent block storage devices to the virtual machines. Essentially the same service as Amazon's EBS. The project was originally a part of Nova (nova-volume), but was separated as a separate project leading up to the Folsom release. The block device can be moved between virtual machines, with the ability to take snapshots and backups. Cinder supports a number of different storage solutions in the back end, ranging from directly attached disks to enterprise solutions from NetApp. The typical application for cinder is database storage volumes, or other disk critical services.

[2]The virtual hard drives are removed when the virtual machine is terminated

Object Storage (Swift). Swift is a "high availability, distributed, eventually consistent object store" [25]. Swift provides an AWS S3 compatible REST API for uploading and retrieving objects into the storage cluster. This API can either be used directly, or through one of the many language-specific libraries that were provided by Rackspace. The typical applications for Swift are document storage and image storage in combination with Glance.

3.4.1.3 Networking

nova-network. Compute provides basic networking configurations through the service nova-network. The default network mode is VLAN Network, where a separate network segment is created for every project. The other configurations are Flat Mode and Flat DHCP Mode where all instances are placed on the same network [26].

Quantum. Quantum provides network connectivity as a service [27] and is a pluggable and API-driven system for managing networks and IP-addresses. Quantum was introduced as a supported core component in the Folsom release, and offers the configuration advanced networking services like Layer 3 routing, QoS and VLANs. Switches and other networking equipment can be managed by Quantum through provided plug-ins.

3.4.1.4 Other services

Dashboard (Horizon). Horizon provides a web-based user interface for the OpenStack services. The interface gives both administrators and end-users easy access to provisioning and controlling virtual machines and storage volumes, and basic user management.

Identity (Keystone). Keystone provides authentication and authorization for the OpenStack services, in addition to a catalog of the services within a particular OpenStack cloud. Both end-users and the other services authenticate against Keystone, and are authorized for different access levels in different projects. Keystone supports a pluggable back-end, including SQL and Lightweight Directory Access Protocol (LDAP), for storing the authentication and authorization data.

Image Service (Glance). Glance provides "discovery, registration, and delivery services for virtual disk images" [28]. The Image Service are most commonly used to store template disk images used to provision virtual machines, but can also be used to store snapshots of running machines. These snapshots can be used for backup

and archiving purposes. Glance can store the images in different back-end storage solutions, including Swift.

3.5 Alternative open source virtualization systems

In this section we will go through two of the alternative open source virtualization systems that are available.

3.5.1 Eucalyptus

Eucalyptus[3] is a widely deployed cloud computing platform providing IaaS. Eucalyptus was founded as a research project at University of California, Santa Barbra in 2007 [29], and was from 2009 until 2011 the main part of Canonicals Ubuntu Enterprise Cloud-project.

Eucalyptus is written in Java and C, and consists of different components that can be deployed on separate machines offering a wide range of services. The different components include: [30]

Cloud Controller (CC). Providing an AWS compatible API for administering the cloud and a web based user interface.

Cluster Controller (CLC). Operating as a gateway between the CC and NC, deciding which node a VM should be deployed on

Node Controller (NC). Controlling the life cycle of the virtual machines and is deployed on all physical machines designated to host VMs. Supports Xen and KVM and VMware.

Walrus. Providing a object store compatible with the AWS S3 API, and can be used to store disk images used for provisioning and snapshots of virtual machines.

Storage Controller (SC). Providing persistent block storage to the virtual machines. Comparable to Amazon's EBS.

[3] `http://www.eucalyptus.com/`

3.5.2 OpenNebula

OpenNebula[4] started as a research project in 2005, and had its first public release in 2008 [31]. Since 2008 the project has had several releases. In 2009 the project got included in the Ubuntu distribution, and later also included in CentOS, Debian and openSuse. The project has a number of contributors from a number of different companies and research institutions, and has an active user community. OpenNebula is written in a wide range of programming languages, naming C, Java and Ruby.

The main feature that differentiates OpenNebula from other projects is that it only requires software on the controller node, as it communicates directly with the hypervisor running on the physical machines used for virtualization. OpenNebula provides support for Xen, KVM and VMware.

OpenNebula consists of the following services: [32]

OpenNebula Daemon. Management daemon controlling virtual machines, users and images.

Match-making Scheduler. Scheduler keeping track of resources, and selecting the most suitable physical machine to deploy a VM.

Monitoring and accounting daemon. Providing statistics and accounting information.

API services. *EC2 Service* for providing an AWS EC2 compatible API and the *OCCI service* for providing compatibility with Open Cloud Computing Interface[5].

Sunstone. Providing a web-based user interface.

As OpenNebula don't provide its own storage solution, it relies on a shared file system for live migrations of virtual machines.

3.6 Summary

In this chapter we have identified requirements for a private cloud deployment in an educational context. This is the fundamental part for the further work in this thesis.

[4] `http://www.opennebula.org/`
[5] `http://occi-wg.org/`

Table 3.7 – Summary of alternative virtualization systems

	OpenStack	Eucalyptus	OpenNebula
Year started	2010	2007	2005
Latest release	2013-04	2013-04	2013-05
Release frequency	~6 months	~4 months	~4 months
Language	Python	Java, C	Java, C, Ruby + others
Hypervisors	Xen, KVM, VMware, Hyper-V, LXC, UML	Xen, KVM, VMware	Xen, KVM, VMware
Deployment	Software installed on all nodes	Software installed on all nodes	Only on front-end nodes
Main storage	Cinder	Storage Controller	Unix filesystem
Object store	Swift	Walrus	N/A
AWS compatible	Yes	Yes	Yes
Licencing	Apache 2.0	GNU GPLv3	Apache 2.0

We have also presented OpenStack, the system to system to be implemented, in addition to briefly presenting alternative solutions. See Table 3.7 for a summary of the different virtualization systems.

Chapter 4

Implementation

This chapter describes the configuration and the choices made when installing and configuring OpenStack, the virtualization system we will be using.

The description provided is on a general level, and not intended as a detailed step by step guide to reproduce the configuration.

Table 4.1 lists the definitions used in this chapter.

4.1 Setup

The setup of OpenStack was done by configuring two physical servers (Table 4.2), one acting as both the controller, and a computing node and the other as a computing node. The two servers were interconnected by gigabit Ethernet.

Table 4.1 – Definitions

Term	Description
VM	A virtual machine
Project	A project, with one or more virtual machines
User	End user, with access to one or more Projects

Table 4.2 – Hardware specifications

Feature	Value
Model	HP ProLiant DL380 G5
CPU	2 x Intel Xeon E5420 2.50GHz Quad Core
Memory	16GB
HDD	2 x 72GB 15kRPM SAS (RAID-1)

The servers were installed with Ubuntu 12.04.1[1] with OpenStack 12.2 (Folsom) packages from the Ubuntu Cloud repository[2]. OpenStack was configured using Puppet[3], an open source configuration management tool, in combination with the official OpenStack-modules[4]. The Puppet manifest used is provided in Appendix A.1.

4.1.1 Networking

OpenStack supports a number of different network layouts. In big scale deployments, hosting different projects or customers, one might want to keep different projects in separated network segments, to avoid potential security breaches and bypassing of firewalls.

In this prototype implementation, with the expected use case of a single student or a group of students needing a single virtual machine, the increased security of using separate networks doesn't weigh up for the added overhead, as this requires changes to the existing network infrastructure.

For this project the network model named Flat Manager is used (See Section 3.4.1.3). This model simply creates a bridge between the physical network and the virtual machines, and automatically assigning them fixed IPs from a range allocated to OpenStack. This model is very similar to the current deployments of other virtualization solutions at IDI.

This configuration places the physical nodes and the virtual machine on the same network. This is not optimal, as the recommended deployment method for OpenStack is to have a separated network for communication between the nodes – system security greatly relies on this separation. See Section 4.1.4 for how we countered these challenges.

[1] Later upgraded to Ubuntu 12.04.2
[2] `https://wiki.ubuntu.com/ServerTeam/CloudArchive`
[3] `https://www.puppetlabs.com/`
[4] `https://github.com/stackforge/puppet-openstack` at commit 02fc41294e

As public IPv4 addresses are a limited resource, the solution with fixed public IPs for the virtual machines also introduces a limitation to the number of VMs. Public IPv6 and NATed IPv4, with floating IPv4 addresses, might be a more suitable solution for a permanent installation.

4.1.2 Hypervisor

OpenStack supports a number of different hypervisors. This is described in Section 3.4.1.1. For this configuration Kernel-based Virtual Machine (KVM)[5] was used, as this is the default on OpenStack installations on Ubuntu, and is thoroughly documented.

KVM is a full virtualization solution for Linux, which benefits from hardware virtualization extensions in the CPUs (Intel VT or AMD-V). The CPUs used in our setup supports 1st generation hardware assisted virtualization from Intel. Being a full virtualization solution, the hypervisor supports virtualizing x86 operating systems without modification. KVM also allows the virtual machines to use paravirtualized I/O device drivers (Virtio), increasing I/O performance [33]. Virtio is supported natively by the Linux kernel, and open source drivers are available for installation on Windows. OpenStack in combination with KVM use Virtio by default for Linux guests.

In addition to virtualizing different operating systems, we can run different versions of the operating systems, allowing us to restore snapshots of archived virtual machines. These are both requirements identified in Chapter 3.

4.1.3 Storage

The initial testing was done using local server storage for the virtual disks and thereby was fairly limited by the 72GB storage capacity on each server.

In the final design of the solution, an iSCSI export from the departments Storage Area Network (SAN) was added. This export was used for storing virtual images. This iSCSI export was mounted on the primary node, and Network File System (NFS) was used to export the file system to the secondary node. This provided a fairly large amount of storage, which could easily be expanded. Logical Volume Manager (LVM) was applied on top of the iSCSI export, to provide additional flexibility for partitioning and expansion of the volume.

[5] `http://www.linux-kvm.org`

The shared file system was used for storing virtual disk images and did not provide persistent storage, as the disk image would be removed when a virtual machine is terminated.

This configuration is not optimal, as all storage traffic to the secondary node will pass through the primary node. This results in additional delay and introduce a single point of failure. Optimally there would be dedicated storage node(s) using Cinder, and separate networking for storage traffic.

4.1.4 Hardening

The recommended solution for deploying OpenStack includes a dedicated network segment for communication between the control node and the compute nodes. In this configuration both OpenStack nodes are on a publicly available network, together with the virtual instances.

One example of an obvious security issue is that the compute nodes will make the VNC-interface for connecting to the instances console available on its public IP.

To counter this we have added firewall rules to the compute node to only allow incoming traffic from the control node. The firewall rules are included in Listing 4.1.

Listing 4.1 – Hardening of compute node

```
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
iptables -A INPUT -p icmp -j ACCEPT
iptables -A INPUT -s $MASTERIP/32 -j ACCEPT
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -j REJECT --reject-with icmp-port-unreachable
```

4.1.5 Virtual machines

Through the imaging service (Glance) the system administrator can provide the users with preconfigured images of virtual machines. For this purpose a default Ubuntu 12.04 image from Ubuntu Cloud Images[6] was provided.

The OpenStack documentation includes a good description on creating and uploading custom images [34]. For a permanent installation one should provide and main-

[6] http://cloud-images.ubuntu.com/

tain a set of different guest operating systems. It is also possible for the users to generate and upload their own images.

The dashboard (Horizon) described in Section 3.4.1.4 provides a web-based user interface, allowing the users to provision and control their virtual machines.

4.1.6 Project quotas

As the system has limited resources, the default quota assigned to the projects were fairly restricted compared to the specifications of a modern physical machine. The quota is shown in Table 4.3.

Table 4.3 – Project quota

Feature	Value
VMs	1
CPU	Single core
Memory	512MB
Disk	10GB
IP	1 (public IP)

4.1.7 Summary

Table 4.4 shows the services running on the two nodes, *vtest01* and *vtest02*. Figure 4.1 shows the logical architecture of the system.

4.2 Monitoring

During the evaluation period the performance metrics of the physical hosts were monitored using Munin[7] and collectd[8], two common resource monitoring tools. The combination of the two were used because collectd provides higher resolution on the data collection (every 10s vs. 5m), while Munin has better support for aggregated graphs.

[7] `http://www.munin-monitoring.org/`
[8] `http://www.collectd.org/`

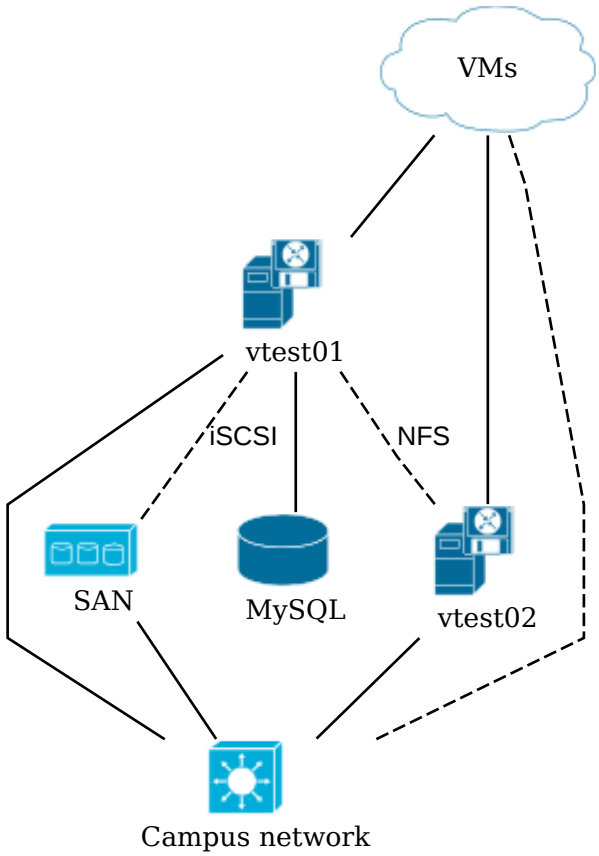

Figure 4.1 – Logical architecture

Table 4.4 – Running services

	Node	
Type	vtest01	vtest02
OpenStack services	nova-api nova-network nova-compute glance-api glance-registry horizon keystone	nova-compute
Additional services	MySQL RabbitMQ NFS server iSCSI client	NFS client

Additionally the usage of Horizon was monitored through the web server's access logs. OpenStack keeps a detailed log of all resource provisioning in its database, making it possible to generate reports of allocated resources.

To ensure that any unscheduled downtime was handled as quickly as possible during the operational period, the service monitor Nagios[9] was configured to do simple service checks. This service monitoring was deployed on an external server, and the description of this is outside the scope of this project. There are multiple Nagios plugins available for monitoring the different components of an OpenStack installation.

4.3 Routines

4.3.1 Managing users

The default model for storing users in OpenStack's identity provider Keystone, is by keeping a user database in SQL. A consequence of this is that you, as an administrator, will have to distribute separate login credentials. This is not very efficient when serving a large user base.

As all of the users of this installation are affiliated with the university, and therefore

[9] `http://www.nagios.org/`

have a user account on the central IT system, we can use LDAP to validate passwords, removing the need for separate credentials. Keystone provides support for LDAP through one of the included authentication drivers. The built-in support requires modification of LDAP schemas, as it also stores information about projects and services. This is not viable, as we do not control the LDAP service.

To counter this we have written a small authentication driver that replacing the password validation against the database with a simple LDAP query. Everything other than the actual passwords will be stored in the SQL back-end. This technique can also be used with Active Directory and other commonly used user databases. The modified authentication driver is attached in Appendix B.1.

4.3.1.1 Adding users

Adding users to the system is done by invoking the adduser script attached in Appendix B.2.

The script is fairly simple, and works by using the NTNU username of the user as an argument. It creates a dedicated project for the user, adds the user and applies the default quota to the project.

4.3.1.2 Removing old users

To avoid that unused machines keep running indefinitely, there should be routines for disabling machines. This can be done by manually disabling all machines after each semester.

In addition you want to avoid using resources on machines owned by students that no longer have active user accounts. This can be done by implementing a script that checks if the account still exists in the central LDAP.

4.3.2 Backup

In every production system, it is important to be able to restore the system to a working state after an incident resulting in loss of data.

In an OpenStack-environment like this, the most significant components are:

- Databases (in MySQL)

- Configuration (in Puppet)
- Virtual machine images

The database contains all metadata about users, projects and virtual machines, and should be backed up on a regular interval. For this installation automysqlbackup[10] was used, in combination with regular system backup.

Virtual machines can be backed up using the snapshot feature in OpenStack. The snapshot feature can also be used to archive virtual machines used for projects that might need follow up at a later time.

4.3.3 Disaster recovery

Like any production system, this system should have routines for disaster recovery. The routines should follow the current practice in the organization.

The most likely risks that must be considered are unplanned power outages and hardware failures, like disk crash or corruption.

The OpenStack Operations Guide [35] contains a number of tips for troubleshooting and recovery from failures, and can be a useful tool for developing such routines.

4.3.4 Maintenance of the system

OpenStack is under heavy development, and even for the stable releases security updates and other fixes are released frequently. All of the components can be updated and restarted without interrupting the running KVM instances. This makes it fairly safe to keep the system up-to-date without scheduling downtime for the users.

In a larger production environment the preferred way to perform maintenance would be to migrate the instances off one node, and upgrade them one by one.

Upgrading to new major releases should be tested in a separate environment, and preferably be performed outside the semester, when the usage is likely to be low.

[10] http://sourceforge.net/projects/automysqlbackup/

Table 4.5 – Fulfillment of primary requirements

Requirement	Fulfilled	Comments
PRQ1	X	Self service provided by Horizon and API
PRQ2	X	Apache 2.0 license [36]
PRQ3	X	No specific hardware requirements
PRQ4	X	Packages available in Ubuntu
PRQ5	X	Active community with many support channels [37]
PRQ6	X	Active development. Over 200 monthly contributors [38]
PRQ7	X	All x86 operating systems supported through the KVM hypervisor

4.3.5 Maintenance of virtual machines

The virtual machines running on the system should be updated regularly with the latest security patches. This could be achieved by enabling automatic updates in the provided guest images.

4.4 Evaluation

Table 4.5 show that all primary requirements (Section 3.2.2) are fulfilled by the system. Out of the secondary requirements (Section 3.2.3) nine out of eleven requirements are fulfilled (Table 4.6). The requirements are summarized in Table 4.7.

Table 4.6 – Fulfillment of secondary requirements

Requirement	Fulfilled	Comments
SRQ1	X	System based on Ubuntu, which is already maintained by the department
SRQ2	X	Implemented through customized authentication module
SRQ3	X	Possible through live migration of virtual machines
SRQ4	X	Provided by Horizon
SRQ5	-	Not possible to ensure that the software is updated
SRQ6	X	Possible through manual routines
SRQ7	X	Possible through snapshots
SRQ8	X	Commercial support available from Canonical [39] and others
SRQ9	X	Nova limits the virtual machines to the assigned IP-address
SRQ10	X	Horizon provides reports of used resources
SRQ11	-	Requires customizing of Horizon

Table 4.7 – Summary of requirements

Type	Fulfilled	Unfulfilled
Primary	7	0
Secondary	9	2
Sum	16	2

Chapter 5

Private cloud in practice

5.1 Methodology

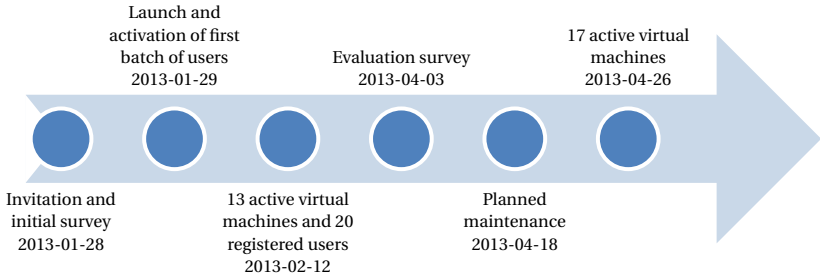

Figure 5.1 – Timeline

In this chapter we will present the experimental phase of this project. The goal for this phase is to see how this system performs in practice, with a real-world workload, and to gain experience about potential problems and weaknesses of the system. Another important goal is to retrieve feedback from users, which can be used for further improvement of the system.

In preparation for the launch, the system was configured as described in Chapter 4. Additionally a Getting Started guide (Appendix C) describing the most common tasks including provisioning and accessing virtual computers was prepared.

5.2 The operational period

The user testing launched January 28, and began with inviting students from IDI to apply for access to the system. The invitation was done by emailing the student lists for the fourth and fifth year Master's students. The testing was limited to this group of students, as the test implementation of the system had limited resources.

During the first day 11 users applied for an account, and an additional 9 users applied during the first week. By the end of the operational period the system had 23 registered users.

In the registration form the users were presented with a survey asking them about their previous experience with virtualization systems, preferred operating systems, and about their expectations to the system. The survey is presented in detail in Chapter 6.

On January 29 the first batch of users were enabled, and an informational email was sent out to the users.

During the period all communication with the users was done using dedicated email lists provided by the technical group. The motivation for doing this was to make a future transition of responsibility for the system to the technical group easier, both by having a fixed contact point for the end users, and by having an archive of all correspondence for future references.

Table 5.1 summarizes the events through the operational period.

5.3 System usage

In this section we will analyze some of the activity data collected through the statistics tools used to monitor the system. The tools used are described in Section 4.2. The timespan for all graphs in this section are from January 29 to May 15 during the operational period.

At most there were 20 active virtual machines running on the system. Figure 5.2 shows a plot of running virtual machines during the period. The maximum of 20 VMs were reached on May 13, this was also the maximum number supported by the system, limited by the number of IP addresses assigned. At the same time there was 13GB of memory and 23 CPU cores allocated to the virtual machines.

Table 5.1 – Operations log

Date	Event
2013-01-28	Invited users
2013-01-29	Added the first batch of users
2013-02-30	Expanded quota for a user that needed additional CPUs
2013-02-02	Added additional users
2013-02-27	Added user
2013-03-08	Created a shared project for two users
2013-04-01	Expanded quota for user with a need for multiple VMs
2013-04-09	Received notification about planned power outage
2013-04-11	Notified the users about planned downtime
2013-04-18	Controlled shut down of the service before planned outage. Used the opportunity to upgrade software on physical hosts
2013-05-02	Added users

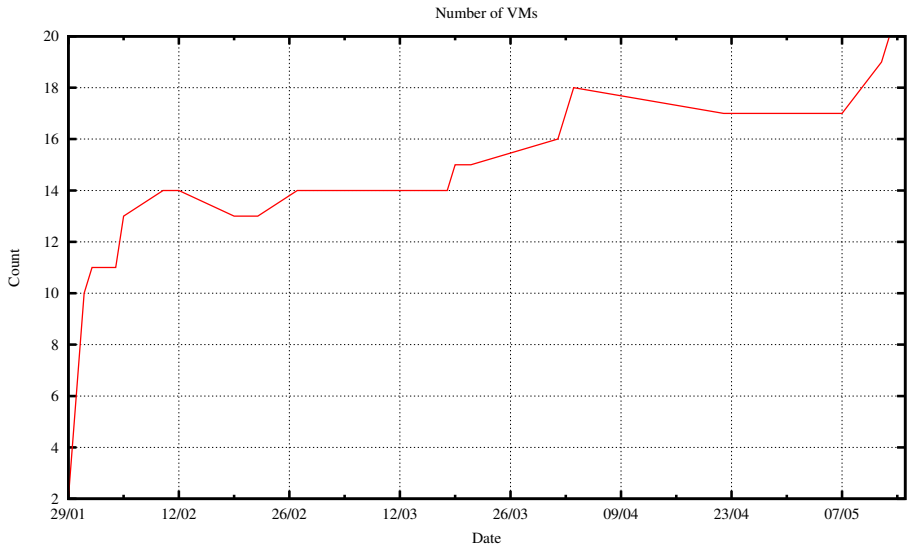

Figure 5.2 – Active virtual machines

The system is primarily managed by the users through the web interface Horizon. Figure 5.3 shows the number of unique visitors per day. These numbers are extracted from the web server access logs, and show an average of about 3.5 visitors per day.

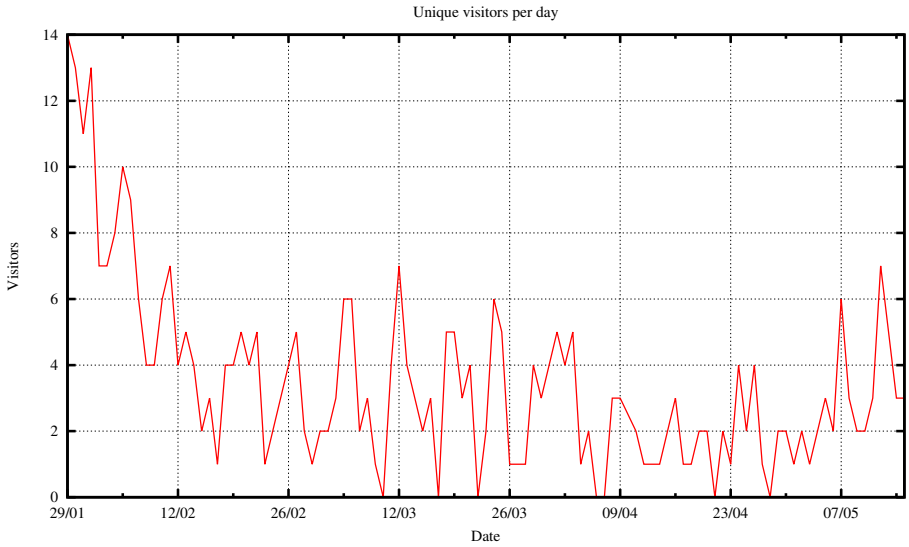

Figure 5.3 – Web interface: Unique visitors per day

Figure 5.4 and 5.5 show the load average [40] for the two physical servers throughout the operational period. The average load on both servers is low, but with fairly high spikes. The peaks are related to high I/O activity when provisioning virtual machines, resulting in high load.

The CPU usage for the nodes is shown in Figure 5.6 and 5.7. The maximum available usage is 800%, as both servers have eight CPU cores. We can see that both servers are mostly idle. However *vtest02* has long periods of usage between 100% and 150%, caused by a virtual machine provisioned on this server which was used for a CPU bound workload in relation to a fellow students master project.

As the host machine needs to keep a copy of all running VMs in memory, the most common limitation when consolidating multiple VMs on a single physical machine is the amount of memory available. The memory usage for our nodes is shown in Figure 5.8 and 5.9. From these graphs we can see that the two servers were far from depleted of available memory during the time span of the operational period. The visible notch in both graphs in Week 16 is from shutting down the VMs in connection to the planned power outage described in Section 5.3.1.3.

We can see that the ”Last” values for used memory (6.6GB and 5.7GB) totals on a value lower than the 13GB of memory allocated to the virtual machines, and that is

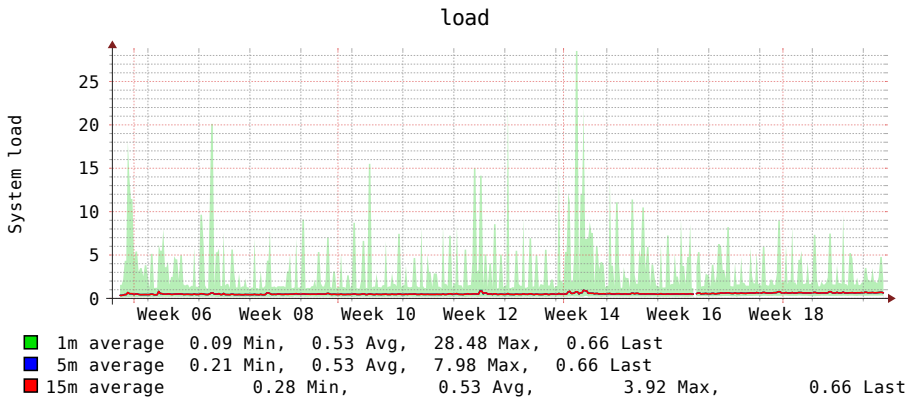

Figure 5.4 – Load average for node vtest01

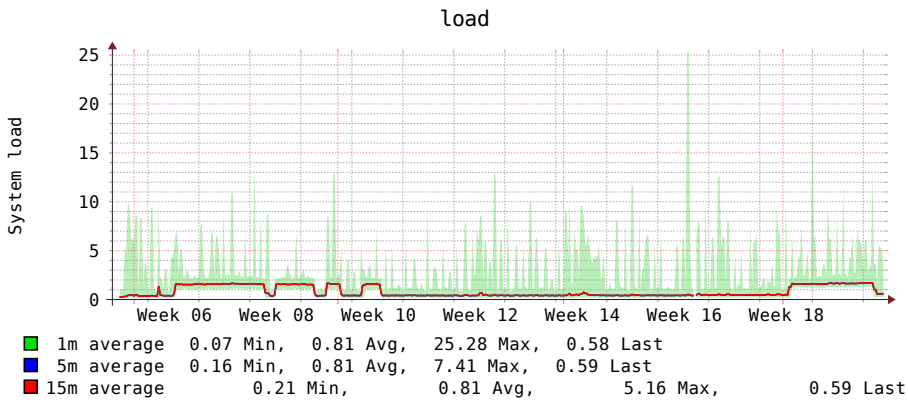

Figure 5.5 – Load average for node vtest02

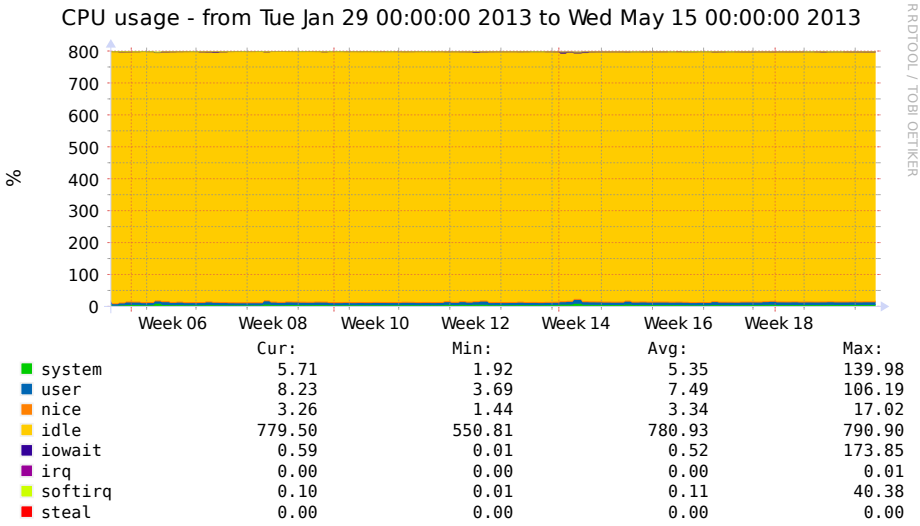

Figure 5.6 – CPU usage for node vtest01

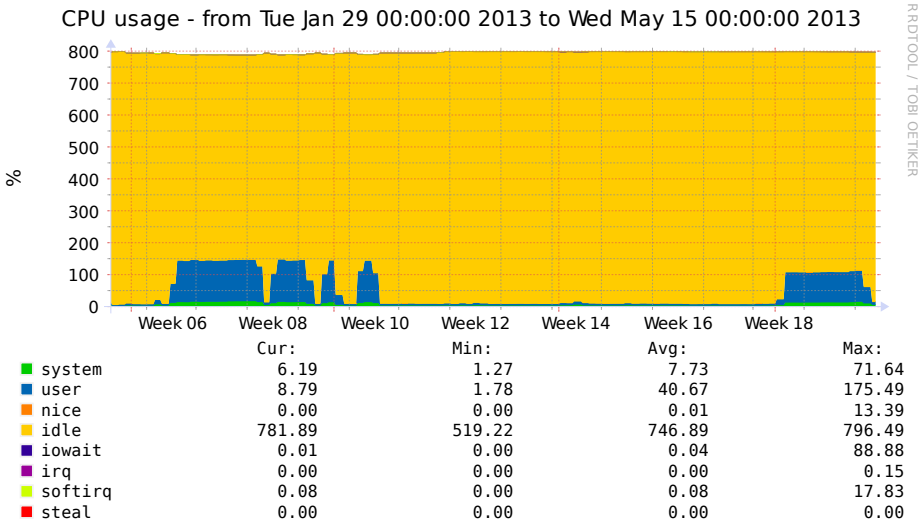

Figure 5.7 – CPU usage for node vtest02

before including the memory used by the software running on the physical servers themselves.

This is mainly because of the memory de-duplication done by KSM [41] for the KVM processes we can see a significant memory saving:

```
root@vtest01:~# cat /sys/kernel/mm/ksm/pages_sharing
719975

root@vtest02:~# cat /sys/kernel/mm/ksm/pages_sharing
536318
```

The page size for a standard Linux system is 4096 bytes, giving us a combined saving of 4.8GB RAM. We see this effect because all 20 virtual machines are running the same operating system, resulting in a very similar memory footprint for all VMs.

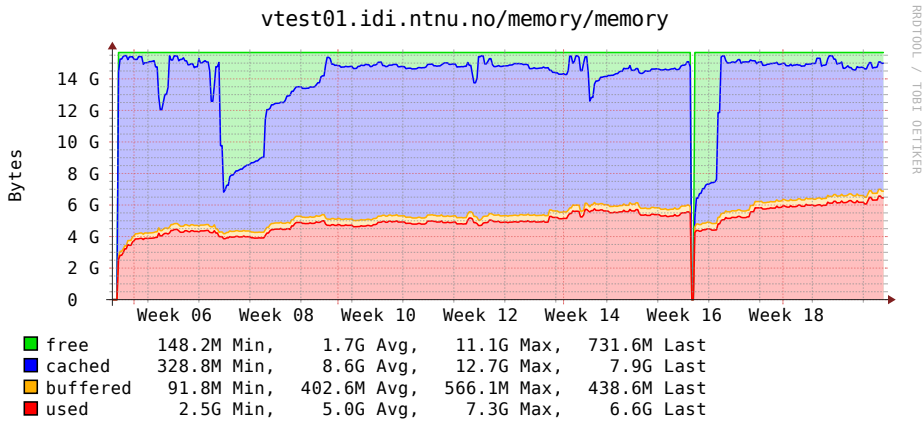

Figure 5.8 – Memory usage for node vtest01

Figure 5.10 and 5.11 show the Ethernet traffic on both physical servers during the production period. The traffic includes both storage traffic (iSCSI traffic between *vtest01* and the SAN, and NFS traffic between the servers). The peaks in network traffic also correlate with the Load graphs, supporting the theory of high load when performing high I/O activity. This can also be seen in Figure 5.12 which shows the traffic to the iSCSI storage.

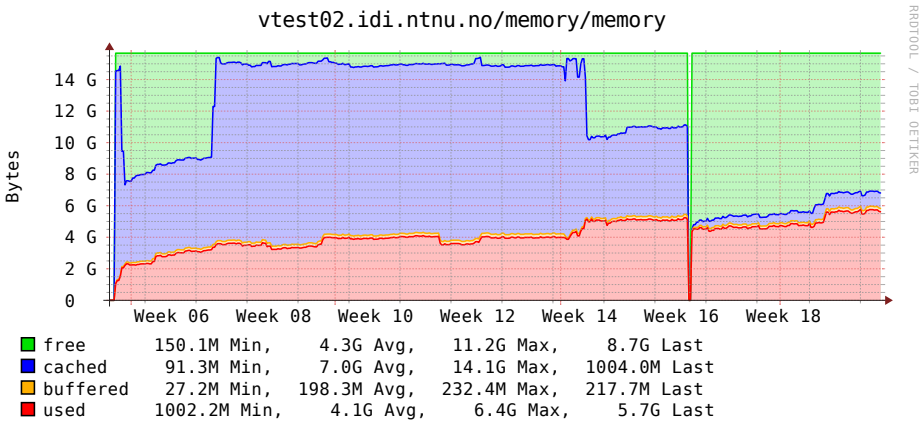

Figure 5.9 – Memory usage for node vtest02

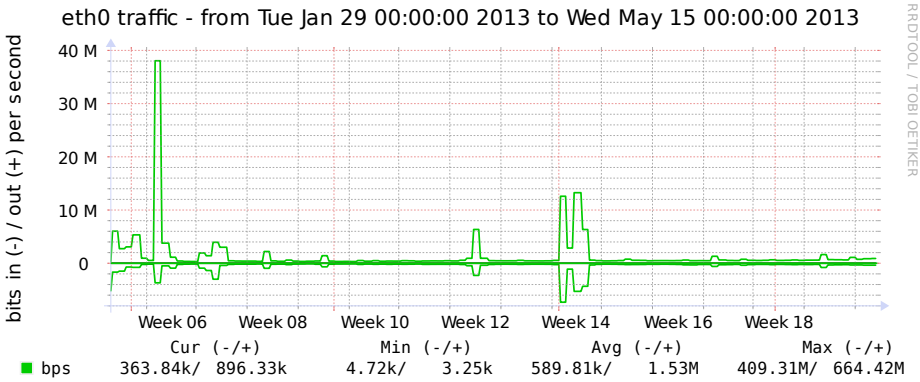

Figure 5.10 – Network traffic for node vtest01

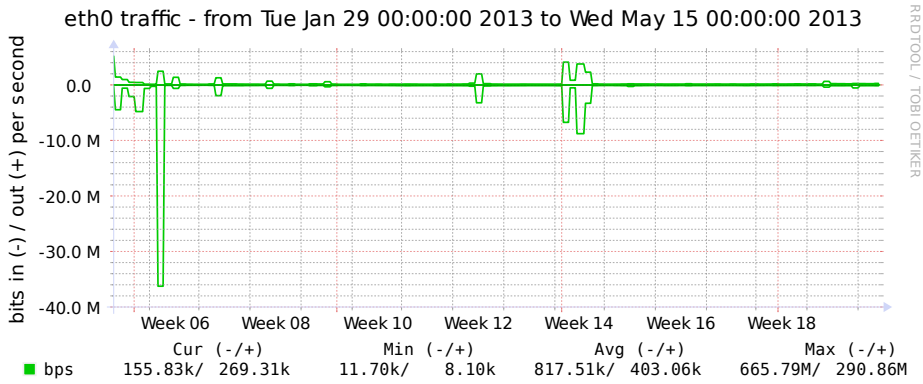

Figure 5.11 – Network traffic for node vtest02

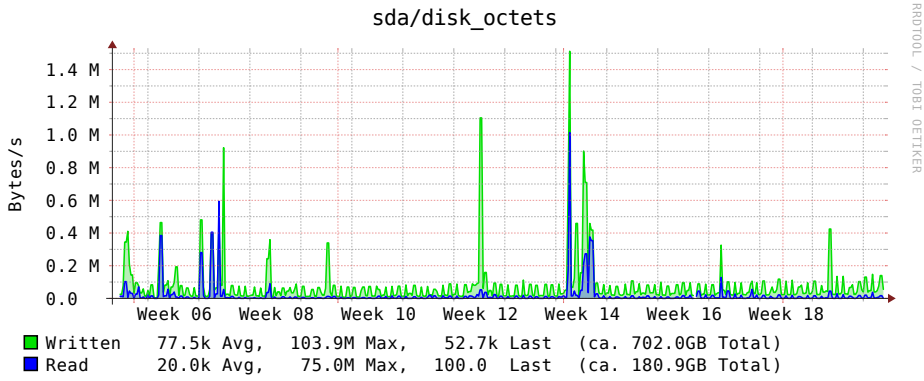

Figure 5.12 – iSCSI traffic

5.3.1 Challenges

5.3.1.1 IPv6 on virtual machines

As the campus network at NTNU is dual stacked, with both native IPv4 and IPv6 support one would want all new machines and services on the network to support IPv6.

Soon after starting testing we noticed a problem with auto assigning IPv6 addresses to the virtual machines:

```
eth0: IPv6 duplicate address 2001:700:300:2101:f816:3eff:fe41:a76
detected!
```

This error is caused due to one of the nwfilter firewall rules in OpenStack. The purpose of this filter is to limit the virtual machines from using IPv4-addresses not assigned to them, but gets an unwanted effect for IPv6.

This is considered as a bug in OpenStack, and a fix is released in the next version. [42]

As a workaround in OpenStack 2012.2 you can disable the duplicate address detection on the virtual machines using the following command:

```
root@ubuntu:~# echo 0 > /proc/sys/net/ipv6/conf/eth0/dad_transmits
```

5.3.1.2 Incorrect quota handling

When the admin user ”Terminate” machines owned by a users Project, the Project will not get the updated quota. To resolve this you need to manually correct the quota in the database.

This is also considered as a bug, and a fix is released in the next version. [43]

5.3.1.3 Planned power outage

On April 9 we received a notification about a planned power outage for the building where the physical servers were located. Shortly after an email was sent to the users, informing them about the scheduled down time.

The day of the outage the servers were manually shut down shortly before the announced time. Before shutting down the servers, all running virtual machines were suspended. Additionally the physical servers were upgraded with the latest software from the Ubuntu package repositories.

When the power was back online the servers were powered back on, and the suspended virtual machines were resumed.

5.4 Evaluation

During the production period, the system performed well, and we did not experience any unscheduled down time. During the experiments we identified two bugs, none which were ”show stoppers”, but this clearly shows that OpenStack is still under development, and emphasizes the importance of keeping the installation up to date. This should be taken into consideration when looking into the possibility for a permanent deployment.

At the most, we had 20 virtual machines running. The usage statistics shows that the system at no time were saturated, and that we could either have increased the default quota, or increased the number of allocated IP addresses and expanded the testing to more students.

From the usage statistics gathered through monitoring the system, we can see that the virtual machines were not used very heavily. However, if we had more active use of the virtual machines, we could easily have ended up with decreased performance. Like shown in the studies performed by Corradi et al. [11].

Chapter 6

Surveys and user feedback

In this chapter we will present the surveys given to the users, and their results. Upon registration, the users were asked to answer a few questions about their expectations to the system, and their expected usage. Later in the semester an evaluation survey was given, asking about the actual use and about the users experience with the system.

These results are not considered representative for the whole student population. For the first survey only 23 answered, and out of these only 16 answered on the follow-up survey. However, the answers and the presentation of data can be a helpful tool for evaluating the success of this and future projects.

The questions were originally given in Norwegian, as this is the native language for most students at NTNU. In this chapter the questions and answers are presented in a translated version. The original Norwegian versions can be found in Appendix D.

Both surveys were conducted using Google Forms, an application in the Google Drive suite.

6.1 Registration survey

In addition to stating their user name, contact information and affiliation, the users were asked to answer some questions about their expected use and expectations when signing up for an account.

The following questions were given upon registration:

- Primary use
- Performance vs. stability
- Requested operating systems
- Previous experience with virtualization
- Expected usage frequency

The goal with these questions was to identify the expectations from the users, and to get input on the expected use.

The different alternatives presented in the survey are listed in Table 6.1. All answers are summarized in Appendix D.1.

Table 6.1 – Registration questions - Translated from Norwegian

ID	Question	Type	Alternatives
Q1	Primary use	Choose one	A1: Master's project A2: Project A3: Other academic purposes A4: Private purposes / try the solution A5: Other
Q2	What is most important of performance and stability on the services you are planning to run on your virtual machine?	Scale (1-5)	1=Performance 3=Equally important 5=Stability
Q3	Which operating systems/Linux-distributions would you like to use through this service?	Multiple choice	A1: Ubuntu A2: Debian A3: CentOS A4: FreeBSD A5: OpenBSD A6: Windows A7: Other
Q4	Do you have previous experience with use or configuration of virtualization software?	Yes / No	-
Q5	What is the expected usage frequency of your virtual machine, and/or the services running on it?	Choose one	A1: Daily A2: Weekly A3: Less than weekly

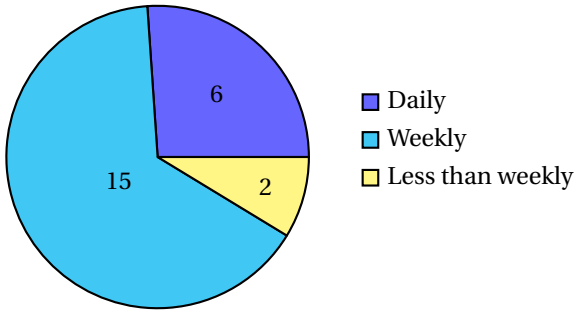

Figure 6.1 – Registration survey (Q5): Expected usage frequency

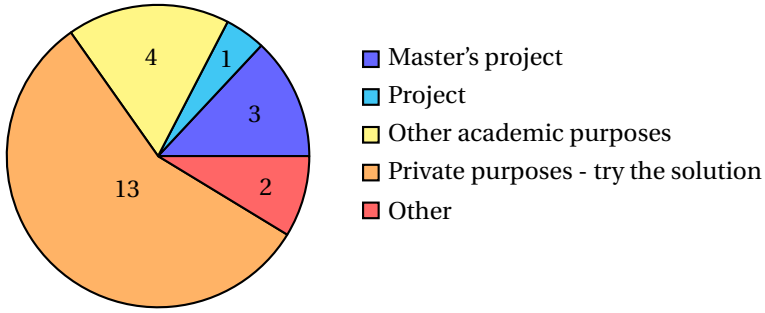

Figure 6.2 – Registration survey (Q1): Primary use of the system

6.1.1 Results

6.1.1.1 Expected usage

In Q2 and Q5, we ask about the users expectations to how much, and in what context they will use this service.

The expected amount of use is fairly high. 21 out of 23 users state that they will use the service weekly, or more. These results are visualized in Figure 6.1.

As shown in Figure 6.2, 13 of the 23 users state that they will use the service for private purposes. The rest of them are divided over "Other" and various academic usage. Both users answering "Other" have stated different academic purposes in a comment.

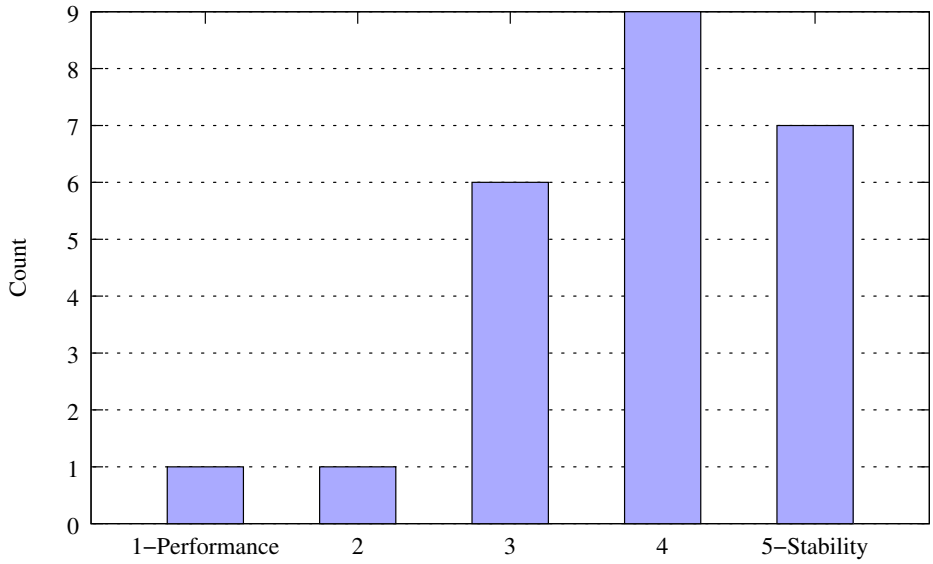

Figure 6.3 – Registration survey (Q2): Performance versus stability

6.1.1.2 Performance or stability

Q2 asks the participant to choose between performance and stability on a range from 1 to 5, where 1 is a high performing (less stable) system and 5 is a stable system with less performance. Most of the answers lean towards stability (Figure 6.3). These results clearly convey what most of the users want from a virtualization system – a stable environment where they can host their application, not necessarily a place to crunch numbers. This is an interesting point to take into consideration when it comes to capacity planning for a larger scale deployment of a virtualization system.

6.1.1.3 Requested guest operating systems

In Q3 the users were asked to name which operating systems they would want the system to provide support for. The most popular variants were the Linux distributions Ubuntu and Debian, closely followed by Windows. The answers are visualized in Figure 6.4.

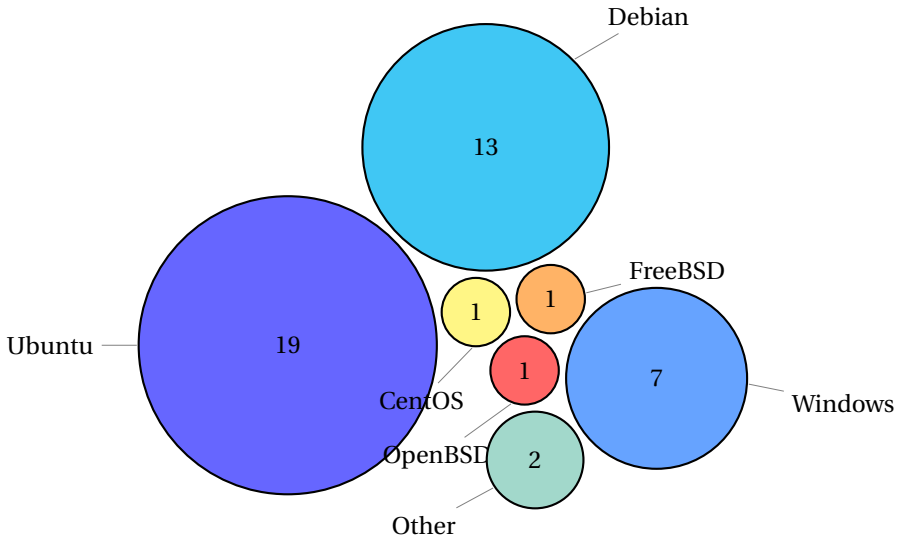

Figure 6.4 – Registration survey (Q3): Requested operating systems

6.1.1.4 Previous experience

In Q4 we ask about previous experience with virtualization software, both with running your own software, or by using public providers.

As shown in Figure 6.5, 15 out of 8 users state that they have previous experience with virtualization software. This is interesting, as it shows that not all of the signed up users are ”enthusiasts” that are early adopters of new technology, and that we have reached out to different types of students.

6.2 Evaluation survey

On April 3 a survey asking about the actual use of the system was sent out to the registered users. The questions are summarized in Table 6.2 and the answers in Appendix D.2. One of the reasons for conducting this survey asking about actual use, is that due to the nature off platform virtualization, we can not assess what's running on inside the virtual computer without advanced analysis outside the scope of this thesis. There are also be privacy issues related to this form of inspection.

Table 6.2 – Survey questions - Translated from Norwegian

ID	Question	Type	Alternatives
Q1	Have you provisioned a virtual machine?	Yes / No	-
Q2	How much did you use your virtual machine?	Choose one	A1: Daily A2: Weekly A3: Less than weekly A4: Once
Q3	What have you used your virtual machine for?	Choose one	A1: Master's project A2: Project A3: Other academic A4: Private purposes / try the solution newline A5: Other
Q4	What alternative would you have used, if this service didn't exist?	Choose one	A1: Existing computer at IDI e.g. workstation at workplace A2: Requested for a new computer at IDI A3: Existing service at NTNU e.g. the common web hotel or Linux login servers A4: External service A5: Private computer A6: Would not have run the application A7: Other
Q5	Has this service made it easier to acquire necessary resources for your work?	Choose one	A1: Yes, it has become easier A2: No, it has become harder A3: No effect
Q6	Has the quota led to any disadvantages or limitations for your usage pattern?	Yes / No	-
Q7	Have you upgraded the software on your virtual machine after installation?	Yes / No	-
Q8	How do you think it was to get started with the service?	Scale (1-5)	1=Hard, 5=Easy
Q9	I have not used this service because ...	Choose one	A1: The user interface was too complicated A2: Did not have the time to try it A3: Did not need a virtual computer
Q10	Is this a service you think IDI should provide on a permanent basis?	Yes / No	-

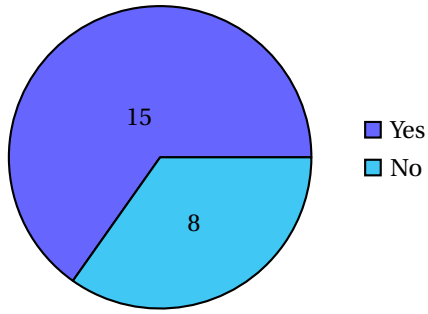

Figure 6.5 – Registration survey (Q4): Experience with virtualization software

6.2.1 Results

Out of the 16 answers, only one replied that he or she had not provisioned a virtual machine. The reason given (in Q9) was that he or she had not found time to try the solution. The users answering ”No” in Q1 were sent directly to Q9, and are the cause of apparent inconsistent sum of answers in this survey.

6.2.1.1 Actual use

To follow up the questions about usage in the registration survey, the users were asked about their actual use of the system. Figure 6.6 shows the results of the actual amount of use (Q2). Most of the users state that they used the system weekly or less.

Figure 6.7 shows the answers for what the users used the system for (Q3). The majority of answers state that they used the system for non-academic purposes.

6.2.1.2 Alternative providers

In Q4 the users were asked where they would have deployed their application if this service didn't exist. Only one of the users replied that he or she would have requested a new computer at IDI. About half of the users replied that they would have used a private computer instead. These answers are shown in Figure 6.8.

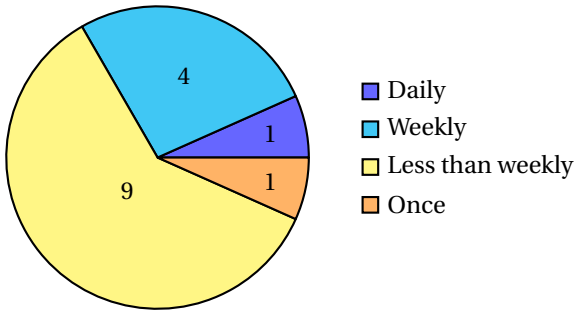

Figure 6.6 – Evaluation survey (Q2): Actual usage frequency

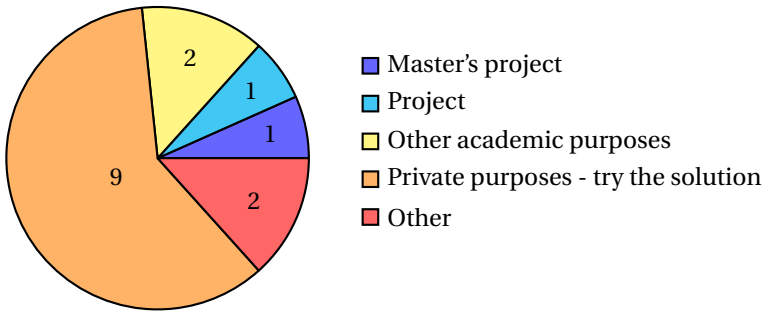

Figure 6.7 – Evaluation survey (Q3): Actual use of the system

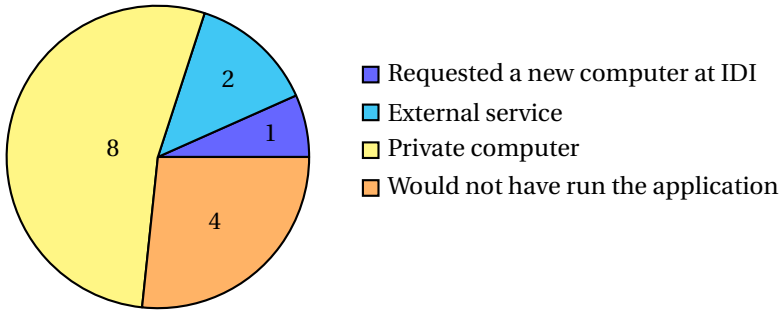

Figure 6.8 – Evaluation survey (Q4): Alternative providers

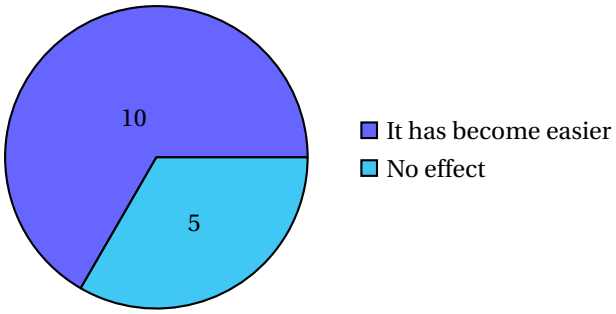

Figure 6.9 – Evaluation survey (Q5): Easier to acquire resources

6.2.1.3 Easier to acquire resources

One of the goals with a private cloud solution is to make computing resources more available to the end users. In Q5 we ask if this solution has made any difference to the way students can acquire necessary resources for their academic work. A majority of the users answered that this has become easier. None of the users answered that that this solution makes it harder to acquire resources. These answers are shown in Figure 6.9. The answers should be seen in the relation of what the users actually used their virtual machines for. As shown in Appendix D.2, three out of the five users answering that this solution had no effect for them also states that they used this service for private purposes. If the solution had more actual academic usage, it is likely that we would have seen an even higher rate of users answering that this solution improves the availability of resources.

6.2.1.4 Quota limitations

The system provides the users with a very limited quota for their virtual machines. In Q6 we ask if this limitation has had any disadvantages for their usage pattern. A majority of the users did not feel that the limited quota had any effect on their use. These answers are shown in Figure 6.10. The results of this question correlate good with what we've seen during the operational period, as a few of the users requested an extended quota for their projects.

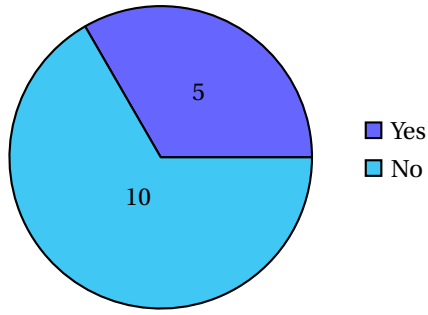

Figure 6.10 – Evaluation survey (Q6): Usage limited by quota?

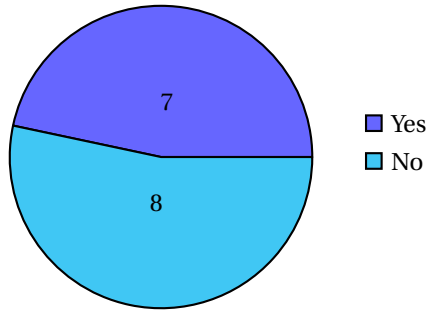

Figure 6.11 – Evaluation survey (Q7): Upgraded VM after installation?

6.2.1.5 Software maintenance

A large number of computers on a network directly connected to the internet might cause a security risk, especially if they are not maintained with the latest security fixes etc. In Q7 we ask if the user has upgraded their virtual machine during the operational period. About half of the users answered ”Yes” to this question. These answers are shown in Figure 6.11.

6.2.1.6 User friendliness

One of the most important success factors of any computer system is that it easy to use. In Q8 we ask the users how it was to get started with the system (Log in, generate keys, configure security groups and provision a virtual machine). The users were

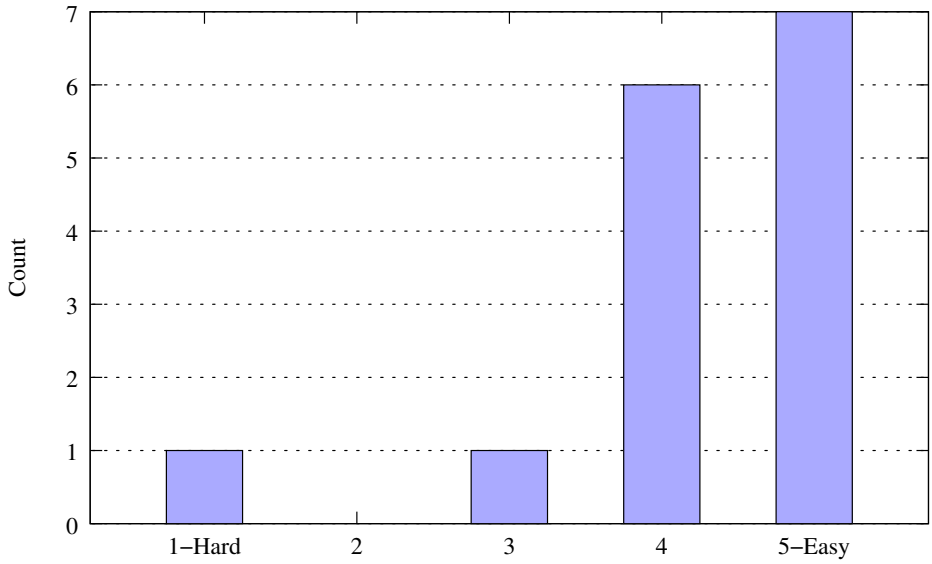

Figure 6.12 – Evaluation survey (Q8): User friendliness

asked to rate the user friendliness from 1 (Hard) to 5 (Easy). The average score at 4.2 shows an overall satisfaction with regards to the user friendliness. The answers are shown in Figure 6.12.

6.2.1.7 Permanent solution

All of the 16 users answered that IDI should develop and provide this service on a permanent basis.

6.3 User feedback

In the evaluation survey, the users were given the opportunity to provide comments and feedback to the system, and further work on the solution. The feedback received is quoted here in a translated version. Original Norwegian quotes are included in Appendix D.2.1.

The feedback shows a general interest in IDI providing a virtualization service to its students.

S1 *Nice service for students without available computer / operating system.*

S2 *The possibility to easily get a number of computers with administrative access to test purposes is a great advantage when developing some types of software. I think this service can be of great help, especially when working with master- or bachelor theses, and I think IDI should consider offering it on a permanent basis in some scale.*

S3 *This is a service that will be very useful during many courses.*

S4 *Great service, I could wish better persistence of the virtual computers for the possibility to run server services etc.*

S5 *This is a fantastic service from IDI. This will make a great difference for those with a need for a server for different purposes. This has been a great help and of great use to me.*

S6 *I think this is a very interesting concept, and I know that many students (including myself) host their own server at home. To have a machine with static IP and with good bandwidth can be very useful for e.g. version control or build servers. Whether IDI should pursue this, I will personally answer yes, but I don't think it should be prioritized, as people usually can find alternative solutions without much trouble.*

6.4 Analysis

6.4.1 Requirements

Through the registration survey we see that the most frequently requested operating systems to be provided by the system are the Linux distributions Ubuntu and Debian, closely followed by Windows.

We also see that most of the users emphasize the importance of stability rather than performance in their virtual machines.

6.4.2 Usage

There is a significant difference in the amount of expected usage and the actual usage of the system. This may sum down to users thinking that they need more resources than they actually do, or that users signed up just to try the solution.

When looking at the area of use, the distribution of academic vs. non academic use is fairly similar in both the expected and the actual usage.

We also see that only half of the users answer that they have upgraded their virtual machine after installation, indicating the need for an automated solution for keeping the VMs upgraded with the latest software / security fixes.

6.4.3 User satisfaction

The users responding to the evaluation survey indicate a high degree of satisfaction with the system. Most of the users think that the system has made it easier to acquire necessary computing resources, and that the default quota provided did not limit their usage. The feedback received as user statements also supports this.

We also see that most of the users are satisfied with the Horizon web interface for controlling the virtual machine.

6.4.4 Demand

From both the survey answers and the user feedback we see a heavy user demand for this type of solution.

However most of the users answers that they have access to alternative solutions for computing resources. These results must be seen in the context of what the service was used for, mostly private purposes.

Even though the students might not *need* this service, we can see from the feedback that all of the responders would like to have the possibility for easy access to computing resources.

6.4.5 Reflection

The evaluation survey was sent to the users on April 3. As shown in Chapter 5 the use of the system increase towards the end of the semester. The answers might have been different if the survey was held at a later time, as some of the users didn't start using the service until after this survey was held.

These results might also be influenced by the fact that only students that were interested in trying the solution have answered. However, we see that only 15 of the 23 students answering on the registration survey have previous experience with virtualization software, which may indicate that we have reached out to a broad set of students.

Although the results we present might not be representative, and the response rate was low, we have chosen to include the diagrams as an illustration to how a future survey can be evaluated.

The evaluation survey was conducted completely anonymously. In retrospect we see that it would have been beneficial if we could track the responses between the surveys, giving us the possibility for a more in depth evaluation of who actually used the service and how it was used.

An alternative method for evaluating the actual need for computing resources could be to read through previous master's theses and project reports at IDI to assess their need for resources based on the nature of their work.

Chapter 7

Conclusion

This chapter gives a summary of each of the objectives described in Chapter 1, and provides a conclusion of the thesis. Finally we give suggestions for future work.

7.1 Data quality

The data collected through our surveys are not to representative, due to a low response rate, and the fact that only students already interested in trying the solution replied. For the evaluation survey the timing was not optimal, as some of the users didn't start using the service until late in the semester. The performance metrics collected through monitoring the system, showing a varying utilization of resources, are likely to be representative for a larger scale implementation.

7.2 Objectives

7.2.1 Identified requirements

As a part of the pre-study to this project we identified key requirements and use cases related to IaaS private cloud implementations in an educational context. During the experimentation we performed user surveys, showing that there is some demand for a solution like this. We also collected important data about what the typical student

user wants from a private cloud solution, namely a stable environment to host their Linux based virtual machine.

7.2.2 Framework for evaluation IaaS deployments

The methodology used when evaluating the system, both through surveys and through monitoring performance metrics can be used as a framework for evaluating systems like the one presented in this report.

All necessary data for reproducing the surveys and system monitoring are included in this report.

7.2.3 Challenges to consider

One of the main complaints against introducing virtualization is the introduction of a single point of failure, unless you use complex high availability techniques and over-provision your resources allowing the system to stay alive with missing compute nodes. These issues are not addressed in this thesis, as the main scope is student projects, and not critical production systems.

As briefly discussed in Chapter 5, high activity in one virtual machine may cause performance degradation for all machines running on the same node. Potential solutions for this are discussed in Section 7.4.

During the operational period of this project we encounter some challenges with the system. Both of these flaws are fixed in the OpenStack project.

7.2.4 Increased availability of resources

The fact that a self-service cloud solution increases the availability of resources to the end user is indisputable. Never before in the history of computing has it been easier to acquire access to computing resources. Our experiment had a focus on students, but the same requirements and principals are also applicable to researchers and other employees at the department.

The benefits from providing a private cloud are two sided. First of all you can provide easy access to resources required by students and researchers to do their academic work. Both for individual work, and as a tool for collaboration in group projects. The

solution can also be used as a platform for conducting practical assignments in many courses.

Secondly, the solution can work as a play ground for students experimenting with private projects. This may facilitate innovation, and will increase the experienced quality of the education.

7.2.5 Potential savings

In this experiment we have successfully consolidated 20 virtual machines on two physical servers. If we consider a consolidation factor of 1:10, this gives us a significant saving in both purchase cost, and in energy consumption.

As shown in this Chapter 5 the consolidation factor could have been even higher, giving even higher savings. Compute nodes with higher performance would also allow a higher density of virtual machines, without increasing the power consumption.

7.3 Conclusion

In this project we have implemented a working prototype installation of OpenStack, and had the system in operation for a period of four months. Even though the resources were limited and the user response was fairly low, we have seen that the system provides a much demanded service among our selection of users, and that further work towards a permanent solution is worth pursuing. Before investing time and equipment in a permanent solution, the department should conduct a larger scale survey to assess the scale of a potential permanent private cloud implementation.

Our experimentation has shown that OpenStack is a suitable system for implementing a private cloud solution, running stable for the whole operational period. Even though we can conclude that OpenStack is suitable for a production environment, we need to emphasize that the project is still under heavy development. You will find bugs, and features may change between the major releases. Some of the bugs can be solved temporally by working around the problem, and most of these will eventually be resolved by the large team of active developers.

Our contribution is a suggested approach for further work towards a permanent private cloud solution at the Department of Computer Science, and in higher education in general. We provide a description of how such a solution can be implemented and evaluated, and highlight challenges that need to be considered.

7.4 Future work

In this section we will present recommendation for further work on this subject.

There are parts of OpenStack outside the scope of this thesis that can be worth researching. Namely the different storage solutions introduced in Section 3.4.1.2 (persistent block storage and the object store) and the possibilities related to using the API for controlling virtual machines.

For user authentication, a Keystone driver integrating with Feide[1] should be implemented. This would benefit both NTNU and other Norwegian educational institutions looking into implementing a private cloud based on OpenStack.

Another approach is to study how OpenStack can be integrated with public clouds, creating a hybrid cloud. This combination is interesting, as this would give the possibility to scale out when needing additional computing resources.

7.4.1 Ideal solution

The ideal deployment of OpenStack should have separate nodes for controllers, storage and compute. Preferably with multiple, high performance compute nodes. The system should provide persistent storage, with the ability to move volumes and access data from multiple virtual machines in the same project. This is possible through Cinder and Swift.

The system should have a delegated subnet of public IPs, and the possibility to configure private VLANs to each project. This can be realized using Quantum, and possibly integrating with Open vSwitch[2].

A set of guest images providing different operating systems and Linux distributions should be maintained by the staff, or by "enthusiast" users. The images could be preconfigured as a generic web server, build server etc. This can also be used by professors or teaching assistants for preparing an environment used in practical exercises.

To avoid high throughput computational tasks interfering with low latency applications, the system should provide different availability zones, allowing users to classify their workload. This can partly be enforced by having different quotas in the different zones.

[1] Centralized identity management solution for the educational sector of Norway, `http://www.feide.no`

[2] Virtual Switch, supporting VLAN isolation, traffic shaping and more. `http://openvswitch.org/`

As seen in the evaluation survey, most users were satisfied with the default quota provided in this prototype implementation. However, an ideal implementation should have a system where the users can apply for an increased quota. This system should also have a minimum of administrative overhead, allowing a ”single click” for both approval and implementation of the updated quota.

Bibliography

[1] Keith Adams and Ole Agesen. A comparison of software and hardware techniques for x86 virtualization. In *ACM SIGOPS Operating Systems Review*, volume 40, pages 2–13. ACM, 2006.

[2] Zach Amsden et al. Vmi: An interface for paravirtualization. In *Proc. of the Linux Symposium*, pages 363–378, 2006.

[3] Rich Uhlig et al. Intel virtualization technology. *Computer*, 38(5):48–56, 2005.

[4] Peter J Denning. Performance Modeling: Experimental Computer Science at its Best. *Communications of the ACM*, 24(11):725–727, 1981.

[5] Gordon E Moore. Progress in digital integrated electronics. In *Electron Devices Meeting, 1975 International*, volume 21, pages 11–13. IEEE, 1975.

[6] Gordon E Moore. Excerpts from a conversation with Gordon Moore: Moore's Law, 2005. Available from: `ftp://download.intel.com/museum/Moores_Law/Video-Transcripts/Excepts_A_Conversation_with_Gordon_Moore.pdf` [cited 13/05/13].

[7] Kevin Lawton. Running multiple operating systems concurrently on an IA32 PC using virtualization techniques. 1999. Available from: `http://web.archive.org/web/20041211213935/http://www.floobydust.com/virtualization/lawton_1999.txt` [cited 13/05/13].

[8] Gil Neiger et al. Intel virtualization technology: Hardware support for efficient processor virtualization. *Intel Technology Journal*, 10(3):167–177, 2006.

[9] David Ott. Understanding VT-c: Virtualization Technology for Connectivity. 2009. Available from: `http://software.intel.com/en-us/blogs/2009/09/30/`

understanding-vt-c-virtualization-technology-for-connectivity/ [cited 30/05/13].

[10] Peter Mell et al. The NIST definition of cloud computing. *NIST special publication*, 800:145, 2011.

[11] Antonio Corradi et al. VM consolidation: A real case based on openstack cloud. *Future Generation Computer Systems*, 2012.

[12] Dylan Steinmetz et al. Cloud computing performance benchmarking and virtual machine launch time. In *Proceedings of the 13th annual conference on Information technology education*, pages 89–90. ACM, 2012.

[13] Åke Edlund et al. Practical cloud evaluation from a nordic eScience user perspective. In *Proceedings of the 5th international workshop on Virtualization technologies in distributed computing*, pages 29–38. ACM, 2011.

[14] Frank Doelitzscher et al. Private cloud for collaboration and e-Learning services: from IaaS to SaaS. *Computing*, 91(1):23–42, 2011.

[15] Lars Erik Pedersen et al. SkyHiGh ADM. 2012. Graduate project. HiG.

[16] Frode Sandholtbråten. Virtualizing terminal rooms. 2008. Specialization project. IDI, NTNU.

[17] Andreas Eriksen. Building a cloud for students at IDI. 2009. Specialization project. IDI, NTNU.

[18] OpenStack Foundation - The OpenStack Blog, 2011. Available from: http://www.openstack.org/blog/2011/10/openstack-foundation/ [cited 30/05/13].

[19] Companies Supporting The OpenStack Foundation, 2013. Available from: https://www.openstack.org/foundation/companies/ [cited 30/05/13].

[20] Releases - OpenStack, 2013. Available from: https://wiki.openstack.org/wiki/Releases [cited 21/05/13].

[21] Stephen J. Vaughan-Nichols. Canonical switches to OpenStack for Ubuntu Linux cloud, 2011. Available from: http://www.zdnet.com/blog/open-source/canonical-switches-to-openstack-for-ubuntu-linux-cloud/8875 [cited 30/05/13].

[22] Welcome to Nova's developer documentation!, 2013. Available from: http://docs.openstack.org/developer/nova/ [cited 21/05/13].

[23] NASA and OpenStack 2012, 2012. Available from: http://nebula.nasa.gov/blog/2012/05/29/nasa-and-openstack-2012/ [cited 21/05/13].

[24] Selecting a Hypervisor - OpenStack Compute Administration Manual - Folsom 2012.2, 2012. Available from: http://docs.openstack.org/folsom/openstack-compute/admin/content/selecting-a-hypervisor.html [cited 21/05/13].

[25] Welcome to Swift's developer documentation!, 2013. Available from: http://docs.openstack.org/developer/swift/ [cited 21/05/13].

[26] Networking Options - OpenStack Compute Administration Manual - Folsom 2012.2, 2012. Available from: http://docs.openstack.org/folsom/openstack-compute/admin/content/networking-options.html [cited 30/05/13].

[27] Welcome to Quantum's developer documentation!, 2013. Available from: http://docs.openstack.org/developer/quantum/ [cited 21/05/13].

[28] Welcome to Glance's developer documentation!, 2013. Available from: http://docs.openstack.org/developer/glance/ [cited 21/05/13].

[29] The Eucalyptus Story, 2013. Available from: http://www.eucalyptus.com/about/story [cited 30/05/13].

[30] Eucalyptus Cloud Components, 2013. Available from: http://www.eucalyptus.com/eucalyptus-cloud/iaas/components [cited 30/05/13].

[31] About the OpenNebula.org Project, 2013. Available from: http://opennebula.org/about:about [cited 30/05/13].

[32] An Overview of OpenNebula 4.0, 2013. Available from: http://opennebula.org/documentation:rel4.0:intro [cited 30/05/13].

[33] Rusty Russell. virtio: towards a de-facto standard for virtual i/o devices. *ACM SIGOPS Operating Systems Review*, 42(5):95–103, 2008.

[34] Creating custom images - OpenStack Compute Administration Manual - Folsom 2012.2, 2012. Available from: http://docs.openstack.org/folsom/openstack-compute/admin/content/creating-custom-images.html [cited 23/05/13].

[35] OpenStack Operations Guide, 2013. Available from: http://docs.openstack.org/folsom/openstack-ops/openstack-ops-manual-folsom.pdf [cited 21/05/13].

[36] OpenStack Open Source Cloud Computing Software, 2013. Available from: http://www.openstack.org [cited 26/05/13].

[37] Ken Pepple. *Deploying OpenStack*. O'Reilly Media, 2011.

[38] The OpenStack Open Source Project on Ohloh, 2013. Available from: http://www.ohloh.net/p/openstack [cited 26/05/13].

[39] OpenStack with Ubuntu, 2013. Available from: http://www.ubuntu.com/cloud/private-cloud/openstack [cited 26/05/13].

[40] Ray Walker. Examining load average. *Linux Journal*, 2006(152):5, 2006.

[41] How to use the Kernel Samepage Merging feature, 2009. Available from: https://www.kernel.org/doc/Documentation/vm/ksm.txt [cited 27/05/13].

[42] Bug #1011134 "hairpin mode on vnet bridge ports causes false positives on IPv6 duplicate address detection". Available from: https://bugs.launchpad.net/nova/+bug/1011134 [cited 06/05/13].

[43] Bug #1078668 "wrong quota_usages updated when admin deletes instance of common user". Available from: https://bugs.launchpad.net/nova/+bug/1011134 [cited 27/05/13].

Appendix A

Configuration

A.1 Puppet manifest

Listing A.1 – ../config/openstack.pp

```
####################
# Node definitions #
####################

node vtest01 inherits openstack_all {
}

node vtest02 inherits openstack_compute {
}

####################
# Common variables #
####################

$public_interface        = 'br100'
$private_interface       = 'br100'
# credentials
$admin_email             = '<email>'
$admin_password          = '<password>'
$keystone_db_password    = '<password>'
$keystone_admin_token    = '<admintoken>'
$nova_db_password        = '<password>'
$nova_user_password      = '<password>'
$glance_db_password      = '<password>'
```

```
$glance_user_password    = '<password'
$rabbit_password         = '<password>'
$rabbit_user             = 'nova'
$secret_key              = '<key>'
$mysql_root_password     = '<password>'
$cinder_user_password    = '<password>'
$cinder_db_password      = '<password>'
$fixed_network_range     = '<network>/<netmask>'
$floating_network_range  = false
$verbose                 = false
$auto_assign_floating_ip = false

$controller_node_address  = '<controller_ip>'
$controller_node_public   = $controller_node_address
$controller_node_internal = $controller_node_address
$sql_connection         = "mysql://nova:${nova_db_password}@${
     controller_node_internal}/nova"

########################################################
# Node providing both controller and compute service #
########################################################

node openstack_all {

class { 'nova::volume': enabled => false }

  include 'apache'
  class { 'openstack::all':
    cinder => true,
    quantum => false,
    public_address          => $ipaddress_br100,
    public_interface        => $public_interface,
    private_interface       => $private_interface,
    admin_email             => $admin_email,
    admin_password          => $admin_password,
    keystone_db_password    => $keystone_db_password,
    keystone_admin_token    => $keystone_admin_token,
    nova_db_password        => $nova_db_password,
    nova_user_password      => $nova_user_password,
    glance_db_password      => $glance_db_password,
    glance_user_password    => $glance_user_password,
    rabbit_password         => $rabbit_password,
    rabbit_user             => $rabbit_user,
    cinder_db_password      => $cinder_db_password,
    cinder_user_password    => $cinder_user_password,
    libvirt_type            => 'kvm',
    floating_range          => $floating_network_range,
    fixed_range             => $fixed_network_range,
    verbose                 => $verbose,
```

```
    auto_assign_floating_ip => $auto_assign_floating_ip,
    network_manager         => 'nova.network.manager.FlatManager',
    allowed_hosts        => [ '<node1>', '<node2>' ],
    mysql_root_password     => $mysql_root_password,
    secret_key              => $secret_key,
    nova_volume        => 'nova',
  }

  class { 'openstack::auth_file':
    admin_password       => $admin_password,
    keystone_admin_token => $keystone_admin_token,
    controller_node      => '127.0.0.1',
  }

}

################
# Compute node #
################

node openstack_compute {

  class { 'nova::volume': enabled => false }

  class { 'openstack::compute':
    cinder => false,
    public_interface   => $public_interface,
    private_interface  => $private_interface,
    internal_address   => $ipaddress_br100,
    libvirt_type       => 'kvm',
    fixed_range        => $fixed_network_range,
    network_manager    => 'nova.network.manager.FlatManager',
    multi_host         => false,
    sql_connection     => $sql_connection,
    nova_user_password => $nova_user_password,
    rabbit_host        => $controller_node_internal,
    rabbit_password    => $rabbit_password,
    rabbit_user        => $rabbit_user,
    glance_api_servers => "${controller_node_internal}:9292",
    vncproxy_host      => $controller_node_public,
    vnc_enabled        => true,
    verbose            => $verbose,
    manage_volumes     => true,
  }
}
```

Appendix B

Code

B.1 Custom authentication driver

This is the Keystone authentication driver used for combining a user database in SQL with external LDAP authentication.

An example of how to use this module with Keystone is included in Listing B.1.

Listing B.1 – Example configuration

```
# /etc/keystone.conf

[identity]
driver = keystone.identity.backends.sqldap.Identity

[ldap]
url = ldaps://at.ntnu.no
user_tree_dn = ou=stud,ou=system,dc=ntnu,dc=no
user_id_attribute = uid
```

Listing B.2 – ../code/sqldap.py

```
# vim: tabstop=4 shiftwidth=4 softtabstop=4

# Copyright 2012 OpenStack LLC
#
# Licensed under the Apache License, Version 2.0 (the "License"); you may
# not use this file except in compliance with the License. You may obtain
# a copy of the License at
```

```
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
# WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
# License for the specific language governing permissions and limitations
# under the License.

"""Identity back-end for Keystone for combining LDAP autentication with
SQL back-end"""

from __future__ import absolute_import

import ldap
from keystone import config
from keystone import exception
from keystone.common import sql
from keystone.common import utils
from keystone.identity.backends.sql import Tenant
from keystone.identity.backends.sql import Role
from keystone.identity.backends.sql import Metadata
from keystone.identity.backends.sql import UserTenantMembership
from keystone.identity.backends.sql import User
from keystone.identity.backends.sql import Identity as SQLIdentity

def _filter_user(user_ref):
    if user_ref:
        user_ref.pop('password', None)
    return user_ref

"""Method that verifies the password using simple_bind to a LDAP server"""
def check_ldappw(user, pw):
    success = True
    user_id_attribute = config.CONF.ldap.user_id_attribute
    user_tree_dn = config.CONF.ldap.user_tree_dn
    try:
        l = ldap.initialize(config.CONF.ldap.url)
        dn = user_id_attribute + '=' + user + ',' + user_tree_dn
        l.simple_bind(dn, pw)
        l.result()

    except ldap.LDAPError, e:
        success = False

    return success

class Identity(SQLIdentity):
    def authenticate(self, user_id=None, tenant_id=None, password=None):
```

```
        """Authenticate based on a user, tenant and password.

        Expects the user object to have a password field and the tenant to be
        in the list of tenants on the user.

        """
        user_ref = None
        tenant_ref = None
        metadata_ref = {}

        try:
            user_ref = self._get_user(user_id)
        except exception.UserNotFound:
            raise AssertionError('Invalid user / password')

        """Hardcoded exception for service users"""
        if user_ref.get('name') in [ 'admin', 'glance', 'nova', 'cinder' ]:
            if not utils.check_password(password, user_ref.get('password')):
                raise AssertionError('Invalid user / password')
        else:
            if not check_ldappw(user_ref.get('name'), password):
                raise AssertionError('Invalid user / password')

        if tenant_id is not None:
            if tenant_id not in self.get_tenants_for_user(user_id):
                raise AssertionError('Invalid tenant')

            try:
                tenant_ref = self.get_tenant(tenant_id)
                metadata_ref = self.get_metadata(user_id, tenant_id)
            except exception.TenantNotFound:
                tenant_ref = None
                metadata_ref = {}
            except exception.MetadataNotFound:
                metadata_ref = {}

        return (_filter_user(user_ref), tenant_ref, metadata_ref)
```

B.2 User management

In this section the script used to create users is included.

Listing B.3 – ../code/adduser.sh

```
#!/bin/bash
```

```
# This script adds a user, creates a project and sets the default quota.
# As the SQL backend requires that we set a password when creating user
# accounts, we generate a random password. This password will never be used,
# when combined with the sqldap Horzion driver.

function usage {
    echo "$0 - Create tenants and users via keystone client"
    echo "Requires SERVICE_TOKEN to be set in environment"
    echo ""
    echo "Usage: $0 username <email>"
    echo ""
    exit 1
}

function get_id () {
    echo $($@ | grep -i " id " | awk '{print $4}')
}

function get_id_list() {
    match=$1; shift
    echo $($@ | grep -i " $match " | awk '{print $2}')
}

if [[ "$1" = "-h" || ! "$#" -ge "1" ]]; then
    usage
fi

USERNAME=$1
EMAIL=$2
PASSWORD=$(pwgen 32 1)

if [[ -z "$SERVICE_TOKEN" ]]; then
    echo "SERVICE_TOKEN not found"
    usage
    exit 1
fi

if [[ -z "$EMAIL" ]]; then
    EMAIL=$USERNAME@stud.ntnu.no
    echo "Using $EMAIL as email"
fi

TENANT_ID=$(get_id keystone tenant-create --name=$USERNAME)
USER_ID=$(get_id keystone user-create \
                            --name=$USERNAME \
                            --pass=$PASSWORD \
                            --email=$EMAIL)

MEMBER_ROLE=$(get_id_list Member keystone role-list)
```

```
keystone user-role-add --user-id $USER_ID \
                       --role-id $MEMBER_ROLE \
                       --tenant-id $TENANT_ID

for quota in volumes:0 \
             gigabytes:10 \
             ram:512 \
             floating_ips:0 \
             instances:1 \
             cores:1; do
    nova-manage project quota $TENANT_ID \
        `echo $quota | cut -f1,2 -d: --output-delimiter=' '` > /dev/null
done
```

Appendix C

Documentation

This appendix show the documentation provided to the end users of the system.

C.1 Getting started

C.1.1 Sign in

After obtaining a user account, sign in at `http://openstack.idi.ntnu.no/horizon` using your NTNU credentials. The system will validate your password using an encrypted LDAP-connection to at.ntnu.no.

C.1.2 Create keypair

To be able to log in to your virtual machines, you need to create (or import) a SSH keypair.

1. Navigate to *Access & Security*
2. Select either *Create* or *Import* Keypair
3. If you choose to create a new keypair, give it a name (e.g. your username) and store the private key in a safe place

4. Make sure that your private key has permission 600 before you use it. Run *chmod 600 filename.pem* in a terminal.

See `http://the.earth.li/~sgtatham/putty/0.58/htmldoc/Chapter8.html#puttygen-conversions` for information about importing your SSH key in putty.

C.1.3 Managing security groups / firewall

By default all incoming connections to the virtual machines are blocked. To open additional ports, you need to add these as rules to your project's security group.

1. Navigate to *Access & Security*
2. Select to create a new security group, or edit the rules in an existing one.

Example: Allow incoming ICMP and ssh traffic from all hosts

IP protocol	From port	To port	Source	Group	CIDR
TCP	22	22	CIDR		0.0.0.0/0
ICMP	-1	-1	CIDR		0.0.0.0/0

Example: Allow incoming http traffic from all hosts

IP protocol	From port	To port	Source	Group	CIDR
TCP	80	80	CIDR		0.0.0.0/0

The updated rule set will be applied on the fly for running virtual machines in that group.

C.1.4 Launch a Ubuntu 12.04 virtual machine

1. Navigate to *Images & Snapshots*
2. Locate the image named *Ubuntu 12.04* and press Launch
3. Give your instance a name
4. Navigate to the *Access & Security*-tab and select a keypair and security group

Important: If you fail to select a keypair, you will not be able to log into your virtual machine

C.1.5 Using your Ubuntu 12.04 virtual machine

1. Navigate to *Instances*
2. Locate your virtual machine and find the IP
3. Log into the machine using ssh as user ubuntu with the key previously imported or generated
4. Use sudo to obtain root access

Example:

```
ssh -i path/to/privatekey.pem ubuntu@<ip>
ubuntu@virtual:~$ sudo -i
```

Appendix D

Survey data

In this appendix the raw data from the surveys are included. The questions were originally given in Norwegian.

D.1 Registration survey

In this section the original Norwegian questions (Table D.1), and all answers (Table D.2) from the registration survey are included.

D.2 Evaluation survey

In this section the original Norwegian questions (Table D.3), and all answers (Table D.4) from the evaluation survey are included.

D.2.1 Statements

In the mid term survey, the users were asked to give feedback in the comments field. The original Norwegian statements are included here:

S1 *Fint tilbud til studenter som ikke har tilgjengelig maskin/operativsystem.*

Table D.1 – Registration questions

ID	Question	Type	Alternatives
Q1	Primært bruksområde	Choose one	A1: Masteroppgave A2: Prosjektoppgave A3: Annet faglig A4: Private formål / prøve ut tjenesten A5: Other
Q2	Hva er viktigst for deg av ytelse og stabilitet på tjenestene du planlegger å kjøre på din virtuelle maskin?	Scale (1-5)	1=Ytelse 3=Like viktig 5=Stabilitet
Q3	Hvilke operativsystemer / Linux-distribusjoner kunne du tenke deg å benytte gjennom tjenesten?	Multiple choice	A1: Ubuntu A2: Debian A3: CentOS A4: FreeBSD A5: OpenBSD A6: Windows A7: Other
Q4	Har du tidligere erfaring med bruk eller oppsett av virtualiseringsløsninger? KVM, VMWare, Amazon EC2 o.l.	Ja / Nei	-
Q5	Hvor ofte antar du at du kommer til å benytte deg av din virtuelle maskin, og/eller tjenesten som kjører på den?	Choose one	A1: Daglig A2: Ukentlig A3: Mindre enn én gang i uken

S2 *Muligheten til å enkelt få et antall maskiner med administratortilgang til testformål er en stor fordel ved utvikling av en del typer programvare. Jeg tror denne tjenesten kan være til stor hjelp spesielt under arbeid med master- og bacheloroppgaver, og mener derfor at IDI bør vurdere å tilby den permanent i en eller annen skala.*

S3 *Dette er en tjeneste som vil være nyttig i svært mange fag.*

S4 *Super tjeneste, kunne ønske meg bedre persistens av de virtuelle maskinene for muligheter til å kjøre servertjenester ol.*

S5 *Dette er en fantastisk tjeneste fra IDI. Dette kommer til å gjøre en stor forskjell for de som har behov for server til diverse formål. Det har vært til stor hjelp og nytte for meg.*

S6 *Jeg synes dette er et veldig spennende konsept, og vet selv at flere studenter (ink. meg selv) setter opp egen server hjemme. Å ha en maskin på fast IP med god båndbredde kan være veldig nyttig for f.eks versjonskontroll og byggeserver. Hvorvidt IDI*

Table D.2 – Registration answers

Timestamp	Q1	Q2	Q3	Q4	Q5
28/01/2013 15:44:56	A1	5	A1	No	A1
28/01/2013 15:45:02	A4	3	A1, A2	Yes	A2
28/01/2013 15:46:11	A4	4	A2	Yes	A2
28/01/2013 15:48:59	A4	4	A1	Yes	A2
28/01/2013 16:27:46	A4	4	A1, A2, A6	Yes	A2
28/01/2013 16:28:03	A4	4	A1, A6, A7	Yes	A1
28/01/2013 17:29:54	A4	5	A1, A2	No	A1
28/01/2013 17:48:22	A3	4	A1, A2	Yes	A2
28/01/2013 18:01:18	A3	5	A1	Yes	A1
28/01/2013 18:09:16	A2	4	A1, A2	Yes	A2
28/01/2013 22:59:25	A3	5	A1, A2, A3, A4, A6	Yes	A2
28/01/2013 23:14:36	A5	4	A1, A2, A6	No	A2
29/01/2013 02:00:09	A4	4	A1	Yes	A2
29/01/2013 12:40:50	A4	5	A2	Yes	A3
29/01/2013 15:18:27	A3	5	A2	No	A2
30/01/2013 15:15:34	A4	3	A1	Yes	A2
30/01/2013 21:50:25	A4	4	A1, A2	Yes	A2
31/01/2013 04:14:56	A1	2	A1, A2, A6	No	A2
31/01/2013 11:22:50	A4	3	A1	No	A2
02/02/2013 22:55:26	A4	3	A1, A5	Yes	A2
27/02/2013 11:34:12	A1	5	A6	No	A1
02/05/2013 23:17:08	A5	3	A1	Yes	A1
05/05/2013 11:44:06	A4	3	A1, A2, A6	No	A3

bør videreutvikle dette vil jeg personlig svare ja for, men jeg synes det ikke burde prioriteres veldig høyt, da folk stort sett kan finne alternative løsninger uten store problemer.

Table D.3 – Survey questions

ID	Question	Type	Alternatives
Q1	Har du opprettet en virtuell maskin?	Ja / Nei	-
Q2	Hvor ofte har du benyttet deg av din virtuelle maskin?	Choose one	A1: Daglig A2: Ukentlig A3: Mindre enn én gang i uken A4: Én gang
Q3	Hva har du brukt maskinen til?	Choose one	A1: Masteroppgave A2: Prosjektoppgave A3: Annet faglig A4: Private formål / prøve ut tjenesten A5: Other
Q4	Om dette tilbudet ikke eksisterte, hvor ville du da ha kjørt applikasjonen din?	Choose one	A1: Eksisterende maskin hos IDI (f.eks. arbeidsstasjon på lesesal) A2: Bedt om ny maskin hos IDI A3: Eksisterende tjenester hos NTNU (f.eks. folk.ntnu.no eller login.stud.ntnu.no) A4: Ekstern tjeneste A5: Privat maskin A6: Ville ikke kjørt applikasjonen A7: Other
Q5	Har tjenesten gjort det enklere å skaffe nødvendige ressurser til arbeidet ditt?	Choose one	A1: Ja, det har blitt enklere A2: Nei, det har blitt vanskeligere A3: Ingen betydning
Q6	Har maskinvarekvoten medført noen ulemper eller begrensinger for ditt bruksmønster?	Ja / Nei	-
Q7	Har du oppdatert programvaren på din virtuelle maskin etter installasjon?	Ja / Nei	-
Q8	Hvordan syntes du det var å komme i gang med tjenesten?	Scale (1-5)	1=Vanskelig 5=Enkelt
Q9	Jeg har ikke tatt i bruk tjenesten fordi ...	Choose one	A1: Brukergrensesnittet var for komplisert A2: Har ikke hatt tid til å prøve den ut A3: Har ikke hatt behov for en virtuell maskin
Q10	Er dette en tjeneste du mener IDI bør videreutvikle og tilby på permanent basis?	Yes / No	-

Table D.4 – Survey answers

Timestamp	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
03/04/2013 12:00:35	Yes	A2	A1	A5	A3	No	Yes	4		Yes
03/04/2013 12:02:38	Yes	A2	A3	A6	A1	No	Yes	5		Yes
03/04/2013 12:07:32	Yes	A2	A4	A4	A1	No	No	5		Yes
03/04/2013 12:09:01	Yes	A3	A5	A5	A3	No	No	5		Yes
03/04/2013 12:11:49	Yes	A3	A4	A6	A3	No	Yes	4		Yes
03/04/2013 12:12:20	Yes	A4	A4	A5	A1	No	No	1		Yes
03/04/2013 12:23:23	Yes	A3	A3	A5	A1	Yes	No	4		Yes
03/04/2013 13:14:50	No								A2	Yes
03/04/2013 14:35:17	Yes	A3	A4	A4	A1	No	No	5		Yes
03/04/2013 15:26:30	Yes	A3	A4	A5	A1	Yes	Yes	5		Yes
03/04/2013 17:44:52	Yes	A1	A5	A6	A1	Yes	Yes	4		Yes
03/04/2013 19:00:05	Yes	A2	A2	A5	A1	No	Yes	5		Yes
03/04/2013 19:08:22	Yes	A3	A4	A5	A3	No	No	3		Yes
03/04/2013 19:42:24	Yes	A3	A4	A5	A1	No	No	4		Yes
03/04/2013 21:04:42	Yes	A3	A4	A6	A3	No	No	5		Yes
03/04/2013 23:43:57	Yes	A3	A4	A2	A1	Yes	Yes	4		Yes