UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science

Valdur Kadakas

Establishing Scientific Computing Clouds on Limited Resources using OpenStack

Bachelor Thesis (6 EAP)

Supervisor: Satish Narayana Srirama, PhD
Co-supervisor: Pelle Jakovits, MSc

Author:.................................... ".....” May 2013

Supervisor:............................... ".....” May 2013

Professor:.................................. ".....” May 2013

TARTU, 2013

Abstract

This work explores how OpenStack cloud platform could be used on limited hardware resources for scientific computing and teaching purposes. OpenStack has deep learning curve and most of the documentation is targeted for creating large scale clouds with hundreds of servers. OpenStack has a lot of components and configuration options which are quite hard to navigate at first. Thus this work tries to provide the rationale for making those technology choices and bases this on sample two server setup belonging to Tartu University Mobile Cloud Lab.

Contents

1 Introduction

OpenStack (1) is one of the fastest developing open source cloud computing platforms. As its name might indicate it is not a single program nor does it have a fixed set of components. It is more like a suite of open source solutions that together form a infrastructure as a service (IaaS) stack. The number of different components to configure and the different configuration possibilities contribute a lot to the complexity of its setup. Learning curve for OpenStack is still deep. There are guides presenting simple demo setups for testing purposes and there is extensive documentation on configuration options but the latter is of more use to a large scale cloud owner who has time and resource to find a configuration suiting their specific needs. This thesis aims to fill part of the gap in between by providing both a sample setup for a small OpenStack cloud deployment running on commodity hardware as well as the rationale for the technology choices made in the process. The work grew out of the needs of Tartu University Mobile Cloud Lab to find an alternative to their existing Eucalyptus (2) installation. They had been using it for research as well as for teaching and were not satisfied with its performance nor stability. They expected a newer open source cloud platform OpenStack to address these issues.

OpenStack is an attractive choice as it is the cloud project with largest momentum at the moment, it is open source, and has the backing of several large organizations and companies including NASA, Rackspace Cloud, Intel, IBM, etc.

The main goal of this work was to provide Mobile Cloud Lab (3) with working installation of OpenStack. They had two servers dedicated to this project and wanted to use this small cloud in the course Basics of Cloud Computing. In case of success, this means that the installation could be used in the course, and with spare time, the cloud was to be extended with 5 PCs formerly used in their Eucalyptus setup. The secondary goal was to run benchmarks, assess this OpenStack installation and try to provide a balanced hardware and software configuration for running scientific calculations in a small private cloud.

Chapter one of this work gives an overview of both OpenStack and Eucalyptus cloud platforms and their components. It also provides a comparison of the two technologies while concentrating on the features relevant for Mobile Cloud Lab. Chapter two describes the requirements of Mobile Cloud Lab, translates these to hardware terms, iterates on likely constraints of given hardware, and sets deployment design principles. Third chapter uses the latter to make technology choices and describes the actual implementation. It also lists software defects that had to be overcome during cloud setup. The fourth chapter presents the results of synthetic performance tests and describes actual performance during Basics of Cloud Computing lab sessions. It also provides advisable amendments to the configuration and gives advise for scaling up the cloud to more hosts.

2

Technology Overview

Everything related to cloud computing has become popular in recent years. The cause is clear as moving IT infrastructure to public clouds allows to dynamically allocate computer resources depening on current needs. Computing is becoming utility in that sense.

It is also possible to establish smaller private clouds. Besides business purposes these are often established in universities. Assoc. Prof. Dr. Srirama of Tartu Univercity has defined the purpose of such clouds in the following way: “With these clouds, researchers can efficiently use the already existing resources in solving computationally intensive scientific, mathematical, and academic problems.” (4)

There are several open souce platforms that allow to establish private clouds. Mobile Cloud Lab of Tartu Univercity has used Eucalyptus technology. An other popular platform is OpenStack. Below we will provides an overview of these cloud platforms and briefly discus their features in the light of using them in teaching and in scientific computing.

2.1 Eucalyptus

Eucalyptus is an open source IaaS cloud platform first released in 2008. Eucalyptus Systems has published their source code under GPL v3 license. They advertise its product as compatible with Amazon Web Services (AWS) APIs. This means that Eucalyptus commands can be used to manage both AWS and Eucalyptus instances. It is also possible to move virtual machine images between Eucalyptus private cloud and

Amazon public one. This allows to create hybrid clouds where applications are set up on private cloud and lend public cloud resources on demand.

Eucalyptus consists of the following components:

- Cloud Controller (CLC): this component provides AWS EC2 functionality, i.e. allows to command virtual server resources
- Walrus: this component provides object storage functionality (S3 in AWS terms) typically used for backup or for storing huge media files
- Cluster Controller (CC): this component provides management service for a cluster in your cloud
- Storage Controller (SC): this component provides block storage functionality (EBS in AWS terms)
- Node Controller (NC): this component controls virtual machine instances

2.2 OpenStack

OpenStack Foundation advertises its cloud platform mainly through openness stating that OpenStack is a result of global collaboration delivering the ubiquitous open source cloud computing platform for both public and private clouds. The project aims to provide solutions for all types of installations. It tries to be simple to implement, yet massively scalable, and feature rich. The software stack consists of a series of interrelated projects delivering various components for a cloud infrastructure solution.

Rackspace Housing and NASA jointly launched this project in July 2010 and released the source code under Apache license. NASA's gave part of the early code from its Nebula and Rackspace from its Cloud Files platforms. As of now more than 150 companies have joined the project (5). This broad industry support indicates that OpenStack is likely to be around for years to come.

OpenStack software is released twice per year. Mobile Cloud Lab's OpenStack deployment was set up using latest stable release at that time - 12.2 named Folsom. This release contained the following main components:

- OpenStack Compute (Nova): is the core of the stack allowing to manage large networks of virtual machines. It makes compute resources accessible via native and Amazon EC2 APIs as well as a web interface. The Compute architecture is designed to scale horizontally on standard hardware. It can work with common virtualization technologies, on bare metal and with high-performance computing configurations. KVM and XenServer are usual choices.

- Object Storage (Swift): is a massively scalable redundant storage system. It allows objects and files to be written to multiple disk drives spread throughout the data center, takes care of automatic data replication and integrity checks. Inexpensive commodity hard drives can be used. Swift is commonly used as a back-end for OpenStack image service Glance.

- Block Storage (Cinder): provides persistent block level storage devices for use with OpenStack compute instances. OpenStack allows cloud user to create, attach, and detach block devices to the servers, i.e. manage their own storage needs. Besides using simple Linux Logical Volume Management it has support for various storage platforms including Ceph, NetApp, Nexenta, SolidFire and Zadara. It is possible to create snapshots of existing block storage volumes allowing an easy way to back ups.

- Networking (Quantum): OpenStack Networking is a pluggable, scalable and API-driven solution for managing networks and IP addresses. It allows to virtualize network stack.

- Dashboard (Horizon): The OpenStack dashboard is a web based administrative interface for cloud resources management.. Its design allows to easily plug in and expose third party products and services like billing, monitoring and management tools.

- Identity (Keystone): OpenStack Identity provides the cloud with central user and access management. It acts as a common authentication system across the cloud operating system and also allows integration with LDAP. Multiple forms of authentication are supported, including standard username and password, tokens, and AWS-style logins.

- Image Service (Glance): stores virtual machine images and snapshots. It can copy or snapshot a server image and immediately store it away. These can be used as a template for getting new server sup and running in very short time frame. Typical work flow consists of configuring one server, creating a snapshots of it and using that snapshot for provisioning multiple servers. It can also be used to store and catalog an unlimited number of backups. Glance can use Swift as back-end.

2.3 Platform Comparison

Both Eucalyptus and OpenStack are open source and offer approximately the same functionality. Eucalyptus is a bit older and won market share due to that. OpenStack started later but has passed Eucalyptus in public interest. The key difference is that Eucalyptus offers a fixed set of components while OpenStack allows to choose from several options. OpenStack is likely to scale better to different needs due to that. It is hard to say which IaasS platform is better. A more appropriate question is which IaaS framework is right for particular application (6).

Mobile Cloud Lab is interested in fast launch time as they need a cloud for cloud computing course lab sessions. There has been research on virtual machine provisioning speed (7) (6) but there is no clear winner. Both platforms are under development and a lot of depends on configuration.

3

Requirements, Constraints, and Design Principles

Tartu University Mobile Cloud Lab needs a small cloud for both teaching and scientific research. They had been using Eucalyptus technology for that purpose before and thus it was quite well defined what was expected of the setup. The new OpenStack setup had to perform better than the existing Eucalyptus one. Existing cloud shortcomings were clear. It could not stand the load that students put on it during lab sessions. It was not known if the cause was in hardware, Eucalyptus software or in the way it was set up but the aim was to address these issues in new OpenStack installation. This cloud platform is more flexible in configuration choices, allows to replace components as the needs change and thus scales better from very small to very large cloud. We will try to define requirements of this cloud installation in both usability and hardware resources terms, list available hardware resources, set design principles for such a small cloud setup, and based on above describe appropriate technology choices for the actual OpenStack installation.

3.1 Cloud Hardware

Eucalyptus was running on five personal computer with each of them having:

- 4 core CPU
- 8GB of RAM

- single 500GB hard disk
- single gigabit NIC

One of the PCs was acting as a cloud controller. The other four were running virtual machine instances. For OpenStack we had two HP ProLiant DL180 G6 servers with each of them having:

- 8 core CPU
- 32GB of RAM
- 2x2TB hard disks
- two gigabit NICs

Those ProLiant servers are more powerful but there are only two of them. As of the Eucalyptus cloud only four hosts was used for running virtual machines those cloud installations would still be comparable in the total number of CPUs and total amount of memory.

3.2 Mobile Cloud Lab Requirements

There would be two typical use cases on cloud installations. Teaching involves two practice sessions as part of two courses Basics of Grid and Cloud Computing and Basics of Cloud Computing where students access cloud services directly. During the two lab session students are to launch new virtual machine, log in over ssh, install Java software, make a snapshot of the running virtual machine, and calculate PI with a Java program. Though OpenStack has a web based graphical user interface (Horizon) students would be using EC2 API either from command line or with browser plugi-ns (hybridfox and elasticfox) for the most of the tasks. As part of the first lab session each student gets its personal user account.

Those tasks above should not cause any severe load on any modern computer but the key problem is that up to 28 students would be doing this in parallel. This had caused the existing Eucalyptus setup crash during the previous years.

Thus clearly set requirements for the cloud installations were:

- withstanding the parallel launch of up to 28 virtual machines in acceptable time frame
- not to crash during all those student operations
- be usable during lab sessions.

Scientific research requirements are different. Typical use case would be running highly parallel computations using MapReduce programming model.

Of course, the cloud software would need to survive provisioning of huge amounts of virtual machines as well as finish the calculations but the response time of the shell of individual instances does not matter that much as nobody would be logging in to all of them at the same time with ssh. Thus this use case did not add any extra requirements to the list above.

In order to find possible hardware constraints on those use cases we need to describe those requirements in hardware terms. In general cloud platform virtualizes four hardware resources:

- cpu time
- memory
- storage
- networking

Any process running on the cloud can be said to become slow if it needs more of those resources than the cloud hardware can provide, i.e. it is waiting for cpu time, its memory pages are being swapped to disk, total storage read/write operations are over available storage bandwidth, or the same about read/write operations over network.

As we are interested in only those cases that cause hardware load we can say that some activity is either cpu intensive, memory intensive, storage intensive (disk reads/disk writes), or network intensive. (We can skip describing activity's memory intensity as we can forbid cloud software from over committing available memory, i.e. launching excess virtual machines)

Virtual machine provisioning would mean getting the machine image from image service Glance (disk reads), transferring it over network (network intensive), storing it on local hard disk (disk writes), creating a copy of it on local hard disk (disk writes) and launching the virtual machine (cpu time and disk writes)

Installing Java involves getting the installer (network, disk writes) and deploying it (cpu and disk writes).

Creating VM snapshot would mean copying it to OpenStack Image Service (Glance) that is both network and I/O intensive on both sides.

Running PI calculations in Java is cpu intensive.

Accessing virtual machine over SSH does not put much load on hardware. What we are interested in is the user experience, i.e. shell responsiveness.

User access over ec2 api and the OpenStack Dashboard can put some load on the cloud components as well if users are constantly trying to check if their virtual machines have become available. Under normal conditions this is unlikely to cause problems but in case of existing load it can make the situation worse.

MapReduce programming model is known for moving a lot of data between the hosts (network intensive) and storing it on local storage (storage intensive). The aim of this model is to run computations in parallel, thus it is cpu intensive as well.

3.3 Likely Hardware Constraints

It was not known to author if given hardware can stand the load that the students are likely to put on it. Mobile Cloud Lab had experienced problems with Eucalyptus on comparable hardware on previous labs but it was not known if the cause was in over committed hardware, Eucalyptus configuration or Eucalyptus software itself.

On software side it is known that both Eucalyptus and OpenStack have had problems with launching a huge number of virtual machine instances in parallel.(6)

Looking at the possible hardware limits it is clear that we can skip the possibility of running out of memory. Those two ProLiant servers have a total of 64GB of memory. Having 28 virtual machines with 2GBs of memory each leaves 4GBs for the operating system and cloud software on both hosts.

CPU resources were an unlikely limit as well. In case of 28 virtual machines there would be less than two instances per core. Running PI calculations would definitely stress the cpus but it is unlikely that all students would run this task in parallel.

Launching virtual machines would also stress the cpus but booting an operating system is something that a modern Xeon cpu can handle with ease.

More likely limits are cloud internal networking and storage.

Launching 28 virtual machines in parallel on two servers would mean copying of the image at least 14 times over a network unless we run Glance on both hosts or cache the images on compute nodes. Both Eucalyptus and OpenStack cache the image if configured properly. Saving snapshots into Glance would also stress the network but this is unlikely to happen in parallel.

Storage bandwidth is the most likely limiting factor on this two server set up. There would be 14 instances per server and with two hard disks this would mean creating 7 virtual machine image files on both of them during launch at best case.

3.4 Deployment Design and Implementation Principles

Deployment of OpenStack is known to be difficult due to the number of different components it may contain and for the number of different configuration possibilities these may have. Mobile Compute Lab does not have human resources to maintain a highly complex cloud installation. This means that setup has to be as simple as possible without losing key functionality. Some optional features of the cloud are not needed. Other areas where we can make compromises include security and high availability.

For the same maintenance reasons this deployment will stick to established and well known technologies if possible. This should ease the learning curve for anyone charged with maintenance work on the cloud software.

This cloud would not be used for running business nor time critical software. There is no serious impact if the the whole cloud is taken down for a day or two for maintenance. Scientific calculations can be restarted and virtual machines and images could be recreated in case of disaster. Detailed and repeatable instructions (in appendix A) allow to set up the cloud from scratch in acceptable time frame. Still, protecting data against simple hardware failure is advisable. Thus, cloud controller data, image files and snapshots, and persistent storage volumes should be on raid1 if possible.

This cloud will not contain confidential information. Of course, only authorized users should have access to the cloud but there is no need to create complex configuration in order to restrict users from seeing each others virtual machines. This cloud will be accessible from campus network only.

There are several tools for automating OpenStack setup but automation is not a goal by itself as there are a few hosts only.

Limited hardware resources will also have an impact on configuration options available. We must set up the cloud on two servers first but must also take into consideration the possibility that we might need to add the five hosts under Eucalyptus to the OpenStack installation later on. Low numbers of hosts means that it is not practical to dedicate whole servers to single specific cloud component being it a cloud controller, image service or persistent storage provider. The lack of high performance file server means that we must distribute I/O operations as evenly as possible between all cloud components. Single NIC gigabit network sets a cap on available network bandwidth. This itself indicates that we can't have dedicated storage servers and must limit network bandwidth if possible.There is no need to create complex configuration in order to restrict users from seeing each others virtual machines. This cloud will be accessible from campus network only.

There are several tools for automating OpenStack setup but automation is not a goal by itself as there are a few hosts only.

4

Deployment Architecture and Implementation

4.1 Technology Choices

There is no universal answer to the questions how to set up a OpenStack. Large production clouds have servers or clusters of servers along with load balancers dedicated to different OpenStack components. A small private cloud does not have those hardware resources nor the need for them as there are less virtual machines and thus less load on storage, on networking as well as on cloud software itself. Its owners are likely to prefer simpler solutions and OpenStack can offer these to them. In fact, one key factor in the growing popularity of this cloud platform is the availability of choice.

In order to create a balanced set up of OpenStack components one must understand how OpenStack and clouds in general use storage and what impact can this usage have on network resources.

4.1.1 Storage

OpenStack as well as Eucalyptus use three types of storage: ephemeral, object, and blocks storage. The latter is also known as volume storage.

Ephemeral storage is allocated when a virtual machine is created. It persists until VM is terminated. VM root resides on it. Besides the root file system it is possible to allocate additional ephemeral block devices to the VM. OpenStack Compute takes care of ephemeral storage. It creates ephemeral storage as files on local server file system under /var/lib/nova/instances directory. This directory should be shared among compute nodes in order to support VM migration. Those files could reside on traditional NFS share or on a distributed file systems like GlusterFS or MooseFS.

As opposed to ephemeral storage object and block storages are persistent. The latter means that they won't be deleted on virtual machine termination.

Object storage is used to access binary objects. Possible operations are upload and download. In place editing is not possible. It is typically used for storing backups and sharing huge media files. OpenStack can store virtual machine images and snapshots in object storage as an alternative to the local file system. Swift is the component taking care of object storage in OpenStack. Third party open source solutions are Ceph's RADOS (8) and Gluster.(9) Both offer distributed object storage with automatic data replication and failover.

Block storage volumes are independent of virtual machine life cycle. Users can create and attach them as unformatted block devices to existing virtual machines. Block storage volumes can be moved between virtual machines but can be attached to only one at a time. OpenStack component Cinder takes care of block storage. It has drivers for several back-ends including Linux LVM and Ceph's RADOS Block Device (RBD) (10).

Ephemeral storage usually resides on local disk and is limited by its bandwidth. It is common to provide faster I/O with block storage by putting it on dedicated storage arrays or on Ceph's RDB cluster.

4.1.2 Networking

Networking decision are an integral part of OpenStack setup. From the guest point of view virtual machines need to talk each other, access remote servers, and be accessible from remote servers. From the cloud perspective OpenStack has to logically separate tenants from each other, allow virtual machines to access remote block storage, copy virtual machine images to compute nodes, allow storage nodes to replicate data over network, etc. There can be a lot of traffic.

OpenStack uses four logical networks: management network for inter-server communication, public network for providing public access to API endpoints, VM network for inter-VM communication, floating IP network for providing public IP accessibility to cloud instances. At best these all should be on separate physical networks in order to avoid congestion. VM and management networks should be separated because of security reasons.

Folsom release offers two networking components. Nova-network is an older and more simpler option and is part of the OpenStack Compute. Quantum is a new and more feature rich virtual network service. Among other things it allows tenants to define their own private networks. Mobile Cloud Lab needs were not complex and thus we don't cover Quantum features in this work.

In its simplest form nova-network running on one of the hosts acts as gateway for all virtual machines. It provides private Fixed IP addresses using dnsmasq (11) to all instances and uses iptables and NAT to connect the VMs to public Floating IP number. Usually the communication is over a separate physical network connecting network host and compute hosts. Both nova-network and all virtual machines use Linux bridging to connect to it.

Though OpenStack recommends at least two separate physical networks between cloud components it is possible to deal with only one.

4.2 A Cloud on Two Servers

Setting up a cloud on two servers is easier than on a cluster of servers. There are still many choices. As stated previously the principles of using standard solutions and avoiding excess components are to be used in current cloud setup. For that reason it is advisable to follow OpenStack official install and deployment manuals (12) unless Mobile Cloud Lab needs dictate otherwise.

Large production clouds are likely to have dedicated servers for different cloud components. Mobile Cloud Lab has two servers only with the possibility that five hosts can be added later on. It is clear that those two servers must distribute all the needed services between themselves initially.

For a minimum production deployment OpenStack recommends to have dedicated cloud controller (running everything but the virtual machines) on a quad core server with 12GBs of memory and compute nodes (running virtual machines) with 32GBs of memory. Having only two servers Mobile Cloud Lab can't afford to have a dedicated cloud controller. Thus the simplest choice would be to make one of them run all the services including virtual machines and the other virtual machines only. Both servers have two network interfaces but cloud controller needs one for Internet connectivity. This leaves us with one physical network for both management and VM network. Two services Cinder and nova-network are likely to cause a lot of traffic if we should expand the cloud. It might be wise to but them on separate hosts.

Before we can decide how to distribute different cloud components between available hardware we need to have a clear understanding what components we need at all. Identity Service (Keystone), Image Service (Glance), and Compute (Nova) are key components that we can't do without. We need Keystone for authentication, Glance for storing virtual machine images and snapshots and Nova for running virtual machines themselves. It is possible to manage virtual machines with API calls only (using OpenStack own API or EC2 API) form command line or browser plug in but OpenStack's native web interface is more convenient for that purpose, thus we need Dashboard (Horizon).

Concerning Object Storage (Swift) we might want to use it as a back end for Glance. Otherwise we don't need it as Mobile Cloud Lab has no plan to directly offer Amazon S3 like service.

Students won't need Block Storage (Cinder) in their labs but persistent storage could be useful for cloud users interested in research as it offers a virtual machine independent storage for data, shell scripts, etc. As it is quite easy to set up in basic form there is no reason not to do it.

OpenStack Networking (Quantum) is a new optional component. Mobile Cloud Lab does not need the advanced virtual networking capabilities it provides and thus it is simpler to use nova-network that is part of Compute.

In the following paragraphs we will describe the choices made in the order in which OpenStack components are typically set up.

OpenStack documentation favors two Linux distributions Ubuntu and Red Hat (along with CentOS and Fedora). They provide deployment guides for both. For a production cloud a release with long term support should be chosen. These are likely to contain OpenStack packages already but given the rapid development of OpenStack its latest stable release should be taken from repositories provided by OpenStack community.

Mobile Cloud Lab opted to Ubuntu taking their latest long term support release 12.04 LTS. As both servers contain two disk we decided to use RAID1 for basic fault tolerance.

OpenStack offers several options for database back end and messaging service. If there is no cause to do otherwise it is advisable to stick to MySQL and RabbitMQ as these are used in official installation manuals. For the same reason it was decided to use Kernel-base Virtual Machine (KVM) for hypervisor.

For simplicity reasons we set Keystone up to use MySQL back-end for user authentication and information storing. Theoretical option would have been connecting Keystone to University LDAP server but this would have added unneeded complexity.

Glance needs a back-end for its images and snapshots. We have two options: either to stores those objects as files on local file system under /var/lib/glance/images/ or use Swift as back-end. We could need Swift for three purposes: for generic object storage (we don't plant to offer this service), for high availability (we are not interested in it), or for distributing load between several servers, i.e. for performance reasons. On current hardware and likely cloud usage pattern there would be no performance gain on image download as all students use the same virtual machine image initially. OpenStack Compute caches it on local file system and thus it is downloaded only once. On saving the snapshot there would be a performance penalty as Swift would replicate it automatically to second server. We are limited in both network and I/O bandwidth and causing additional network traffic and disk writes is not a good idea. It makes sense to protect Glance image store from hard disk failure, i.e. use raid1.

Virtual machine instance image type can have a large effect on both instance launch time as well as run time I/O performance. In general the cloud administrator has to choose between fast launch time and better run time performance. Several Nova options allow to tune this (13) and the choice should depend on hardware. The VM image in Glance should preferably be in qcow2 format. It support compression and thus makes copying to the compute node light on network. By default Nova converts the downloaded file to raw format and extends it to requested size. There is one time performance penalty as Nova caches both the original raw image as well as the extended version of it. After that Nova creates a qcow2 image of it for the actual virtual machine. Qcow2 format is the default as it allows to provision VM very fast because only changes from the original image get written to this instance image. Using raw images for instance disks might provide better concurrency as all VMs would be operating on their own copy. Still the default Qcow2 format is preferable to Mobile Cloud Lab needs as it allows to reduce I/O operations on VM launch.

There is no other option but to store ephemeral storage on server local disk. Using shared storage would allow virtual machine migration but would likely put a double I/O load on the disks in our case. With NFS the server exporting NFS share would have double load with the other server being idle. NFS could be a solution with dedicated storage server with large disk array.

Another commonly alternative to NFS for ephemeral storage is GlusterFS. It provides fault tolerance as all locally written VM instance files would be automatically replicated to other servers in the cluster. This works well with large number of hosts as only 2 or 3 replicas are needed. With two servers this would double the I/O load as all changes would need to be written on both servers. Besides disk I/O this would also put a load on networking. With local storage with two disks there are three options: either to use raid1, raid0 or none. From performance perspective raid1 would speed up reads, raid0 both reads and writes. We start with raid1. In case of poor performance we can switch to raid0.

There are several options for volume storage back-end. Most of these are related to third party commercial storage solutions. OpenStack's own component Cinder, formerly known as nove-volumes - is an iSCSI solution that uses Logical Volume Manager (LVM) for Linux. Besides Cinder other interesting open source options are GlusterFS and Ceph RADOS block device which both scale out, offer redundancy and high availability. For a two server cloud these are clearly an overkill to implement. That leaves us Cinder. Due to limited hardware we can offer only data persistence and not higher speed. We will set Cinder up on cloud controller and put the LVM volumes on raid1 as we did with Glance.

There are actually two choices with nova-network. You are to use VLAN Network mode if you need to separate cloud projects from each other. Otherwise you can stick to Flat DHCP Network Manager which runs dhcp server (dnsmasq) and gives virtual machines their IP numbers from predefined subnet. Mobile Cloud Lab used the latter option.

4.3 Implementation

This chapter describes what was actually done to set up the cloud on those two servers. This deployment was largely based on OpenStack Install and Deploy Manual Ubuntu for the Folsom release. OpenStack setup is pretty straightforward if everything is set up exactly the same way as in the deployment manual. Problems arise if one has different hardware or needs to configure something differently. We will list only major choices, configuration differences and issues that came up during setup. Detailed and repeatable instructions answering the question how are part of Appendix A.

We chose Ubuntu 12.04 LTS server edition for the operating system. Deployment was straightforward server base install plus ssh daemon. We partitioned two 2TB disks manually into 128GB raid1 set for operating system and 1TB raid1 set for OpenStack. The latter was for ephemeral storage and for Glance image store. Separate raid set allowed us to switch to raid0 in case of performance problems without repartitioning the drives. Third raid set (raid1) was on controller only and for 500GB Cinder persistent storage LVM physical volume. Part of the drives was left unpartitioned for flexibility.

After reboot we updated installed software and added Ubuntu Cloud Archive repository in order to get OpenStack Folsom release packages.

Following the guide we deployed NTP, MySQL, and RabbitMQ.

Deployment guide provides a shell script for populating Keystone DB schema. The script needed a bit of tuning and unfortunately is not safe to rerun several times. It took some time to figure out why services did not start to work as expected. It is not easy to adjust already set values. Manually updating a few MySQL table values solved part of the problem but in the end recreating Keystone database in MySQL and rerunning the script was the solution.

Next component was Glance. As said we did not deploy Swift and set Glance up to use local file storage under /var/lib/glance.

As hinted before we created separate partitions for /var/lib/glance and /var/lib/nova. We allocated 250GB for both and left the rest of the 1TB physical LVM volume vacant. The cause was that we had no idea how much space would Glance and image instances occupy.

Cinder is actually a code branch of nova-volume. It is possible to use either of them with Folsom release. There is no functionality difference. Following the guide we ended up using nova-volume. It has to be replaced with Cinder as nova-volume will be deprecated in the next release.

Though most of the changes in OpenStack configuration involve passwords and IP number changes it is a good idea to back up original files. This way it is easier to understand what was changed in case of encountering problems. The configuration samples in Folsom deployment guide were not all up to date. There were some deprecated values and some mistakes. Fortunately OpenStack log files stored in /var/log/ alerted us and allowed to sort these issues out with the help of documentation.

Major problem was with nova-networking. First, the network configuration sample presented in deployment guide presumes that there exist separate physical networks for cloud management and for inter-VM traffic. It is possible to join them into one physical network but deployment guide does not hint if this is possible or how to do it. Sample configuration with single physical network is in Appendix A. In case of networking problems time invested in understanding Linux bridging and nova-networking is worth it (14).

Second, nova-network was first set up on cloud controller. Later on there was need to move it to the other node as Floating IP range got routed to its public interface. Unfortunately nova-network did not start up on it properly and virtual machines did not get their IP numbers. Exploring configuration, logs, and source code lead to Keystone database tables content in the end. IP network was connected to particular nova-network instance. OpenStack documentation is of little help if something is not done by the book.

The last OpenStack component to deploy was Horizon (Dashboard). Besides configuring virtual host in Apache the only change done was adding Mobile Cloud Lab to the welcome text.

In addition to OpenStack components we also deployed monitoring software Munin.

4.4 Software Defects

OpenStack is still a new project and not everything is working as expected. We encountered at least three serious defects which we had to find a workaround.

Regular security upgrade that we installed introduced a fix to security vulnerability adding quota to Fixed IPs. Existing project that did not have the quota set used the default value of 10. As a result it was not possible to launch more than 10 virtual machines per project. To make things worse it was not possible to adjust the quota in Dashboard nor with the command line tools. This bug (15) was addressed quickly but still we had to back-port the fix to get the cloud usable.

Other two issues came up during Cloud Computing lab sessions. First, one private snapshot some how denied the listing of Glance images and thus launching new virtual machines. Temporary work around was to make this snapshot public.

During heavy load nova-network failed to remove iptables chain for one virtual machine it had terminated. This locked up related Fixed IP number in such an way that though OpenStack could allocate it to a new virtual machine and it was possible to access it with this IP number it denied access to this virtual machine with any attached Floating IP. Deleting the chain resolved the issue. There are reports that both Eucalyptus and OpenStack are affected by this issue as both use iptables and bridging for attaching Floating IP numbers to VM (6).

5

Performance Tests and Results

The main goal of this work was to deliver a working cloud platform for the cloud computing course labs. The final tests were the labs sessions of course, but before taking the cloud into use we had to verify that the hardware and software is up to the task. Of course we had launched a few virtual machines manually to see if the cloud software was working at all and had tested the functionality that was supposed to work. But a more systematic approach was needed.

Manual functionality tests passed, i.e. it was possible to do everything that the students were supposed to do in the labs.

5.1 Synthetic Performance Tests

In order to assess the impact a full lab of students would have on the hardware we handpicked two tasks that were supposed to stress the hardware most. These were launching virtual machines and deploying Java software.

5.1.1 Virtual Machine Launch Time

First we measured the launch of 28 virtual machines in parallel. About 25 students were suppose to attend the lab. 28 is a bit more but still less than 2 virtual cpus per actual processor core. We used Ubuntu 12.4 server image and launched the virtual machines with 1 cpu and 2GB of memory. Maximum memory consumption would have been 28GB leaving 4GB to the operating system and cloud software. Thus swapping was not expected.

Measuring launch time was straightforward. We knew at what time we had started the process. We can say that Ubuntu server is running when we can log in with ssh. Ubuntu logs all the boot activity including launch of SSH daemon to /var/log/boot.log. All we had to do is to log in with ssh and take the timestamp of this file. The time difference gives us the needed launch time.

Launch of single VM took less than two minutes. Parallel tests with 28 VMs gave us results between below 7 minutes. The differences between boot.log timestamps showed us that first VMs game up in about 2 minutes before the last ones. These numbers are acceptable from user experience perspective.

As part of virtual machine launch Nova had created and grown the qcow2 image files to 203MB. This makes about 3GB of disk writes per server.

5.1.2 Java Install Time

Next we measured Java JRE install time. In order to exclude software download time we created a snapshot of a virtual machine where we had downloaded JRE related packages but not installed yet and used it to launch new virtual machines. On a single host JRE install took little over 3 minutes.

We used parallel-ssh to run the deployment in parallel on all 28 virtual machines taking the timestamp before and after the execution of the program. Of course we had to verify that Java had actually been deployed.

Several tests run returned results around 24 minutes. Monitoring tool top reported varying load peaking at more than a hundred. Top revealed that cpu time was spent on iowait. JRE install grew virtual machine image files from around 300 to 670 MB. This makes about 6 GB per server. It is not much in size but it consists of random writes mainly. It was clear that underlying storage could not handle such disk activity with ease.

5.2 Actual Performance in Labs

15 to 20 students attended lab sessions at a time. In general this cloud installation was usable. Mobile Cloud Lab team member confirmed that this new cloud platform was more stable than Eucalyptus one had been during the previous two year. Still there were issues. Though there were less students than there were virtual machines

in the synthetic test they did not start their VMs exactly at the same time. Some still launching their VMs while others were installing Java already. Server load was less in numbers but for a prolonged time. Munin charts later on revealed that ephemeral storage average I/O write time had been 5 seconds at its maximum. With cloud controller having high I/O writes already 20 Dashboard sessions automatically polling server list twice a minute can put an additional load on it. Dashboard became unresponsive in the end. Being under time pressure we started Apache to get the Dashboard working again. Unfortunately we were not able to make sure the root cause. My personal hypothesis is that all Apache thread were tied up waiting response from Keystone or some other OpenStack component.

5.3 Likely Amendments to Configuration

Test results confirmed what was suspected from the beginning. Storage bandwidth is the most likely limiting factor on this two server cloud setup. Hardware can not put up with the load that students cause when they launch that many virtual machines at the same time. Too many instances are starting up per each hard disk.

We had set up Ephemeral storage on software raid1. Replace mirroring with raid0 should double possible I/O bandwidth. This is the only option available with current hardware.

If two disks in raid0 do not provide enough bandwidth adding a third and forth one could help. An other and likely better hardware upgrading path is to add a solid state drive (SSD) and put ephemeral storage on it . These drives are small but expensive yet provide very good random write performance. Those 14 virtual machines per server used up only 10GB of disk space.

We had allocated 250GB to both Glance and Nova for ephemeral storage. To our surprise Glance had used up 62 per cent of it by the end of lab sessions. Snapshots created in lab sessions make the most of it. In case of Nova image cache is filling the disk. Those are these snapshots that every student created and that they used to launch new virtual machines. On one server cache has grown to 213GB. This is a known issue that Nova allows image cache to grow until it fills the disk. This means that we need to enlarge these file systems or clean the cache regularly.

5.4 Extending Cloud to 10 Servers and up

With one or two or three servers, all of them are likely to be used for running virtual machines. At some point as the number goes up one server should be dedicated to cloud controller functions. Of course larger cloud puts more load on the controller but as we saw from our tests running virtual machines on the controller host can reduce the responsiveness of cloud software. Load issues with Dashboard would not have happened if virtual machines had not used up all the available I/O bandwidth.

Likely services to move out of controller node as the cloud grows are Swift or Cindel. The latter can live on dedicated storage array until it runs out of bandwidth or space. Logical step forward is to replace it with Ceph on three servers. By that time Swift should run on multiple nodes as well.

Ephemeral storage can be local or shared. The latter allows to migrate instances between servers. In a cloud like the one of Mobile Cloud Lab with heavy peak I/O on ephemeral storage it is hard to see any benefits in having a shared storage on a dedicated storage array. It is hard to match the total I/O of all compute node disks. A better solution seems to be speeding up local I/O with raid0 or raid10 and use them as building blocks for shared storage with a distributed file system like GlusterFS or MooseFS.

6 Conclusions

The goal of this work was to provide Mobile Cloud Lab with working installation of OpenStack. They had two servers dedicated to this project and wanted to use this small cloud in the course Basics of Cloud Computing. We achieved the goal as the new installation proved to be more stable than their former Eucalyptus platform.

We successfully predicted that this setup could have problems with storage I/O. During lab sessions students are to start in parallel almost the maximum number of virtual machines the cloud could run. This puts enormous load on server local storage during VM provisioning. Tests and user experience during lab sessions confirmed this. As a result we advised to double local storage bandwidth by switching the file system under ephemeral storage from mirroring to raid0.

Current work showed that OpenStack is capable platform and it is possible to create a production setup with rather limited hardware resources. It also became clear that in this kind of setup the cloud owner should count the maximum number of virtual machines that are likely to be launched at the same time when planning a new installation. Storage bandwidth must match this load.

Concerning future work it might be interesting to research alternative options for speeding up ephemeral storage in the cloud. Those 5 PCs that were part of Eucalyptus installation still need adding to OpenStack. This would allow to test how putting ephemeral storage onto distributed file systems like GlusterFS and MoosFS (16) would impact virtual machine provisioning times.

7

Sisukokkuvõte

Teadusarvutusteks mõeldud vähese jõudlusega pilvedel loomine OpenStacki näitel

OpenStack on hetkel kõige kiiremin arenev vabal tarkvaral põhinev pilveplatvorm maailmas. Juba nimi viitab sellele, et tegemist ei ole ühe programmiga. Tegemist on pigem vabal tarkvaral põhinevate projektide kogumiga, mis koostöös võimaldavad pakkuda riistvara infrastrukuuri ühtsa teenusena. OpenStacki komponentide arv ning valikute hulk teeb selle seadistamise üsna tülikaks. Juhised pakuvad küll välja lihtsaid demolahendusi ning kirjeldavad kõikvõimalikke konfiguratsioonivalikuid, kuid kõik see on pigem suunatud suure pilve omanikule, kel on aega ja raha leidmaks üles just neile sobiv konfiguratsioon. Käesolev töö püüab täita vahepealse tühja ala pakkudes välja väikese pilve jaoks sobiva näidislahenduse ning kirjeldades ning põhjendades ka pilve ülesse seadmise käigus tehtavaid valikuid.

Antud töö kasvas välja Tartu Ülikooli Mobiilipilve labori vajadusest uue pilve järele. Varasem Eucalyptuse platvormil olnud lahendus oli liiga aeglane ega hiilanud ka stabiilsusega. Valik langes OpenStackile, sest selle taga on hetkel kõige rohkem arendajaid ning ka suurfirmade tugi.

Töö peamine eesmärk oli saada püsti uus OpenStacki tarkvaral jooksev pilv, mis vastaks Mobiilipilve labori vajadustele. Selle jaoks oli ette nähtud kaks serverit ning pilve plaaniti kasutada nii õppetöös kui ka teadusarvutusteks. Edu korral oleks tulnud sellele pilvele lisada ka Eucalyptusest üle jäänud viis lauaarvutit, kuid sellega polnud

kuigi kiire. Edu oleks seisnenud muidu selles, et antud pilve peal oleks saanud läbi viia pilve aluste aine praktikumid. Töö teine eesmärk seisnes pisut suuremale pilvele tasakaalus konfiguratsiooni välja pakkumises.

Pilve tööle saamine õnnestus õigel ajal ning praktikumid sai läbi viia OpenStacki peal. Uus pilv osutus palju staabiilsemaks, kuid esines kas probleeme. Antud pilve eripära on see, et praktikumide käigus tõmmatakse korraga käima väga palju virtuaalmasinaid. Nii nagu oli ennustatud tekitasid need serveri kõvaketastele väga suure koormuse. Virtuaalmasinad läksid vähehaaval küll käima, kuid pilve kasuajaliides Horizon oli väga aeglane. Lahendusena pakub autor välja serverite ketaste ribalaiuse kahekordistamise. Hetkel asuvad virtuaalmasinate failid raid1 peal, kuid need saab tõsta ümber raid0 peale.

Antud töö näitas, et ka piiratud resurssidega on võimalik kokku panna täiesti töötav pilv. Selgus ka, et selliste pilvede puhul, kus korraga ajatakse käima väga palju virtuaalmasinaid, tuleks tõsiselt arvestada kiiremate andmekandjate vajadusega.

Bibliography

[1] Openstack foundation (May 2013).
URL `http://www.openstack.org/` 1

[2] Eucalyptus (May 2013).
URL `http://http://www.eucalyptus.com/` 1

[3] Mobile cloud lab (May 2013).
URL `http://mc.cs.ut.ee/` 1

[4] S. Srirama, O. Batrashev, E. Vainikko, Scicloud: Scientific computing on the cloud, in: CCGRID '10 Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, IEEE Computer Society Washington, DC, USA 2010, 2010. 3

[5] Companies supporting the openstack foundation (May 2013).
URL `http://www.openstack.org/foundation/companies/` 4

[6] G. von Laszewski, J. Diaz, F. Wang, G. Fox, Comparison of multiple cloud frameworks, in: R. Chang (Ed.), IEEE CLOUD, IEEE, 2012, pp. 734–741. 6, 10, 21

[7] D. Steinmetz, B. W. Perrault, R. Nordeen, J. Wilson, X. Wang, Cloud computing performance benchmarking and virtual machine launch time, in: SIGITE Conference, 2012, pp. 89–90. 6

[8] Rados (May 2013).
URL `http://ceph.com/ceph-storage/object-storage/` 14

[9] Glusterfs (May 2013).
URL `http://www.gluster.org/` 14

[10] Rdb (May 2013).
URL `http://ceph.com/ceph-storage/block-storage/` 14

[11] Dnsmasq (May 2013).
URL `http://www.thekelleys.org.uk/dnsmasq/doc.html` 15

[12] Openstack installation guide for ubuntu 12.04 (lts) (May 2013).
URL `http://docs.openstack.org/grizzly/openstack-compute/install/apt/content/` 15

[13] Openstack libvirt images (May 2013).
URL `http://www.pixelbeat.org/docs/openstack_libvirt_images/` 17

[14] Openstack networking tutorial: Single-host flatdhcp-manage (May 2013).
URL `http://www.mirantis.com/blog/openstack-networking-single-host-flatdhcpmanager/]` 20

[15] Bug: Fixed ips quota can break upgrades (May 2013).
URL `https://bugs.launchpad.net/nova/+bug/1161190` 21

[16] Moosefs (May 2013).
URL `http://http://www.moosefs.org/` 26

Appendix A

Cloud Controller Installation Notes

```
### UBUNTU INSTALL ###
{

ubuntu 12.04.1 server base install + sshd
manual partition setup on 1 of 2 hard disks
{ partion table (640GB of 2TB allocated):
(parted) p
Model: HP LOGICAL VOLUME (scsi)
Disk /dev/sda: 3906963632s
Sector size (logical/physical): 512B/512B
Partition Table: gpt

Number  Start       End          Size        File system  Name  Flags
 1      2048s       4095s        2048s                          bios_grub
 2      4096s       503807s      499712s     ext4
 3      503808s     250503167s   249999360s                     lvm
 4      250503168s  1250502655s  999999488s                     lvm
}
LVM and file systems:{
/boot 250MB ext4
VG stratus 128GB:
 32GB LV swap
 32GB LV /root ext4
VG nova-compute 512GB:
}

no automatic updates
server name: stratus.at.mt.ut.ee


simple firewall configuration:{
ufw allow sshd
ufw enable
}

update os packages {
aptitude update
aptitude full-upgrade
reboot
aptitude purge linux-image-3.2.0-29-generic
}
}
```

```
### FOLSOM aka 2012.2 INSTALL ###

### ntp, MySQL, RabbitMQ, Folsom repository install ###
{

install ntp{
apt-get install -y ntp
sed -i 's/server ntp.ubuntu.com/server ntp.ubuntu.com\nserver 127.127.1.0\nfudge 127.127.1.0 stratum 10/g' /etc/ntp.configuration
service ntp restart
}

install MySQL{
apt-get install python-mysqldb mysql-server
# set mysql root password to "BFtPtJxr"
sed -i 's/127.0.0.1/0.0.0.0/g' /etc/mysql/my.cnf
service mysql restart

# sercure mysql
/usr/bin/mysql_secure_installation
}

Installing RabbitMQ{
apt-get install rabbitmq-server
}

Add Folsom repository{
echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-updates/folsom main" > /etc/apt/sources.list.d/folsom.list
apt-get install ubuntu-cloud-keyring
aptitude update
}

}

### Keystone Setup ###
{
apt-get install keystone
# delete sqlite database
rm /var/lib/keystone/keystone.db

mysql -u root -p

mysql> CREATE DATABASE keystone;
mysql> GRANT ALL ON keystone.* TO 'keystone'@'%' IDENTIFIED BY '****';
mysql> GRANT ALL ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY '****';
mysql> FLUSH PRIVILEGES;


/etc/keystone/keystone.conf{
# replace "connection" line with
connection = mysql://keystone:****@193.40.36.71/keystone
# set admin_token
admin_token = ***********
}

# restart keystone
service keystone restart

#initialize the new keystone database, as root
keystone-manage db_sync

# download https://github.com/openstack/keystone/blob/master/tools/sample_data.sh,
# rename it to populate_keystone.sh, and edit to your needs
diff sample_data.sh.orig populate_keystone.sh {
50a51,59
> CONTROLLER_PUBLIC_ADDRESS="193.40.36.71"
> CONTROLLER_ADMIN_ADDRESS="193.40.36.71"
> CONTROLLER_INTERNAL_ADDRESS="193.40.36.71"
> ADMIN_PASSWORD="****"
> DEMO_PASSWORD="****"
> SERVICE_PASSWORD="****"
> ENABLE_SWIFT="yes"
> ENABLE_QUANTUM="yes"
> ENABLE_ENDPOINTS="yes"
80a90,95
> DEMO_PASSWORD=${DEMO_PASSWORD:-$ADMIN_PASSWORD}
> if [[ "$DEMO_PASSWORD" == "$ADMIN_PASSWORD" ]]; then
>     echo "The default demo password has been detected.  Please consider"
>     echo "setting an actual password in environment variable DEMO_PASSWORD"
> fi
>
113c128
<                                          --pass="$ADMIN_PASSWORD" \
---
>                                          --pass="$DEMO_PASSWORD" \
}
# run script:
bash /root/populate_keystone.sh {
+-------------+---------------------------------------------------------+
|   Property  |                          Value                          |
+-------------+---------------------------------------------------------+
|   adminurl  | http://193.40.36.71:$(compute_port)s/v1.1/$(tenant_id)s |
|      id     |             1f4c64b23e784c12b65dc5ec6c8183c3            |
| internalurl | http://193.40.36.71:$(compute_port)s/v1.1/$(tenant_id)s |
|  publicurl  | http://193.40.36.71:$(compute_port)s/v1.1/$(tenant_id)s |
|    region   |                        RegionOne                        |
|  service_id |             a80d2474c9ca4c78bf0ebca5fc5edded            |
+-------------+---------------------------------------------------------+
+-------------+-----------------------------------------+
|   Property  |                  Value                  |
+-------------+-----------------------------------------+
|   adminurl  | http://193.40.36.71:8773/services/Admin |
|      id     |     f541117e371c4f18ba8d07a17f083073    |
| internalurl | http://193.40.36.71:8773/services/Cloud |
|  publicurl  | http://193.40.36.71:8773/services/Cloud |
|    region   |                RegionOne                |
|  service_id |     1586610f130740e4823ee61be8847865    |
+-------------+-----------------------------------------+
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |   http://193.40.36.71:9292/v1    |
|      id     | ac9406541ba040adbebbd33d7a4b809c |
| internalurl |   http://193.40.36.71:9292/v1    |
|  publicurl  |   http://193.40.36.71:9292/v1    |
|    region   |            RegionOne             |
|  service_id | e50d224872a54a7d97b3079558531402 |
+-------------+----------------------------------+
+-------------+------------------------------------------+
|   Property  |                  Value                   |
+-------------+------------------------------------------+
|   adminurl  | http://193.40.36.71:$(admin_port)s/v2.0  |
|      id     |     581b011d3e694a4689fcb5ca9a7c99dd     |
| internalurl | http://193.40.36.71:$(public_port)s/v2.0 |
|  publicurl  | http://193.40.36.71:$(public_port)s/v2.0 |
|    region   |                RegionOne                 |
|  service_id |     292cf744b7f94cef9ce55a54b31ef5d1     |
+-------------+------------------------------------------+
+-------------+-------------------------------------------+
|   Property  |                   Value                   |
+-------------+-------------------------------------------+
|   adminurl  | http://193.40.36.71:8776/v1/$(tenant_id)s |
|      id     |      e990f69421e343ddbf56c39af30bee67     |
| internalurl | http://193.40.36.71:8776/v1/$(tenant_id)s |
|  publicurl  | http://193.40.36.71:8776/v1/$(tenant_id)s |
|    region   |                 RegionOne                 |
|  service_id |      1eefaea0554044349bbb30c7d06126d3     |
+-------------+-------------------------------------------+
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |       OpenStack Dashboard        |
|      id     | 3159bcd43f0c43569d01de98204609ad |
|     name    |             horizon              |
|     type    |            dashboard             |
+-------------+----------------------------------+
+-------------+------------------------------------------------+
|   Property  |                     Value                      |
+-------------+------------------------------------------------+
|   adminurl  | http://193.40.36.71:8080/v1/AUTH_$(tenant_id)s |
|      id     |        3998f1009629445299bc220f105d15d9        |
| internalurl | http://193.40.36.71:8080/v1/AUTH_$(tenant_id)s |
|  publicurl  | http://193.40.36.71:8080/v1/AUTH_$(tenant_id)s |
|    region   |                   RegionOne                    |
|  service_id |        f6bef9ed93ee4cd0b5b5d1c3e9b3fb27        |
+-------------+------------------------------------------------+
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |     http://193.40.36.71:9696     |
|      id     | 8990ebc52d0d423ab391c52af3bf7245 |
| internalurl |     http://193.40.36.71:9696     |
|  publicurl  |     http://193.40.36.71:9696     |
|    region   |            RegionOne             |
|  service_id | 0a9bcf6336cd41259883bcb352912460 |
+-------------+----------------------------------+
}

# add .keystonerc for convenience
cat ~/.keystonerc
export OS_USERNAME=admin
export OS_PASSWORD=****
export OS_TENANT_NAME=demo
export OS_AUTH_URL=http://localhost:35357/v2.0
# source the file
valdur@stratus:~$ . .keystonerc
# test for isntance with
valdur@stratus:~$ keystone token-get
valdur@stratus:~$ keystone user-list

}

### Compute and Image services ###
{
apt-get install glance
rm /var/lib/glance/glance.sqlite

????
10/10/12: When using the Ubuntu Cloud Archive, you need to re-install the python-keystoneclient after installing the glance packages listed above,
otherwise you see an error.
????

mysql -u root -p {
mysql> CREATE DATABASE glance;
mysql> GRANT ALL ON glance.* TO 'glance'@'%' IDENTIFIED BY '*****';
mysql> GRANT ALL ON glance.* TO 'glance'@'localhost' IDENTIFIED BY '****';
mysql> FLUSH PRIVILEGES;
mysql> quit
}

some backups before configuration:{
cd /etc/glance/
root@stratus:/etc/glance# cp -p glance-api.conf glance-api.conf.orig
root@stratus:/etc/glance# cp -p glance-api-paste.ini glance-api-paste.ini.orig
root@stratus:/etc/glance# cp -p glance-registry.conf glance-registry.conf.orig
root@stratus:/etc/glance# cp -p glance-registry-paste.ini glance-registry-paste.ini.orig
}

diff glance-api.conf.orig glance-api.conf{
49c49
< sql_connection = sqlite:////var/lib/glance/glance.sqlite
```

```
---
> sql_connection = mysql://glance:****@193.40.36.71/glance
314,316c314,316
< admin_tenant_name = %SERVICE_TENANT_NAME%
< admin_user = %SERVICE_USER%
< admin_password = %SERVICE_PASSWORD%
---
> admin_tenant_name = service
> admin_user = glance
> admin_password = ****
320c320
< #config_file = glance-api-paste.ini
---
> config_file = /etc/glance/glance-api-paste.ini
326c326
< #flavor=
---
> flavor=keystone
}

diff glance-api-paste.ini.orig glance-api-paste.ini{
57a58,61
> # added by Valdur
> admin_tenant_name = service
> admin_user = glance
> admin_password = ****
}

service glance-api restart

diff glance-registry.conf.orig glance-registry.conf{
28c28
< sql_connection = sqlite:////var/lib/glance/glance.sqlite
---
> sql_connection = mysql://glance:****@193.40.36.71/glance
74,76c74,76
< admin_tenant_name = %SERVICE_TENANT_NAME%
< admin_user = %SERVICE_USER%
< admin_password = %SERVICE_PASSWORD%
---
> admin_tenant_name = service
> admin_user = glance
> admin_password = ****
80c80
< #config_file = glance-registry-paste.ini
---
> config_file = /etc/glance/glance-registry-paste.ini
86c86
< #flavor=
---
> flavor=keystone
}

# not 100% sure of the following need
diff glance-registry-paste.ini.orig glance-registry-paste.ini{
19a20,22
> admin_tenant_name = service
> admin_user = glance
> admin_password = ****
}
service glance-registry restart

NB! after looking at glance loggs had to set paste.ini files with full paht in conf files.

glance-manage version_control 0
glance-manage db_sync
service glance-registry restart
service glance-api restart

# Note
```

```
This guide does not configure image caching, refer to http://docs.openstack.org/developer/glance/ for more information.

# Glance verification with test image
mkdir /tmp/images
cd /tmp/images/
wget http://smoser.brickies.net/ubuntu/ttylinux-uec/ttylinux-uec-amd64-12.1_2.6.35-22_1.tar.gz
tar -zxvf ttylinux-uec-amd64-12.1_2.6.35-22_1.tar.gz

cat .ostackrc {
export OS_USERNAME=admin
export OS_TENANT_NAME=demo
export OS_PASSWORD=****
export OS_AUTH_URL=http://193.40.36.71:5000/v2.0/
export OS_REGION_NAME=RegionOne
}
. .ostackrc

# verify by uploading image
{
glance image-create \
--name="tty-linux-kernel" \
--disk-format=aki \
--container-format=aki < ttylinux-uec-amd64-12.1_2.6.35-22_1-vmlinuz

glance image-create \
--name="tty-linux-ramdisk" \
--disk-format=ari \
--container-format=ari < ttylinux-uec-amd64-12.1_2.6.35-22_1-loader

# take kerne_id and ramdisk_id from previous outputs
glance image-create \
--name="tty-linux" \
--disk-format=ami \
--container-format=ami \
--property kernel_id=386652ae-a92f-48fc-b3ec-84850ebf4ae1 \
--property ramdisk_id=3abc2f1f-ca96-490d-9e14-3cc5e10eb685 < ttylinux-uec-amd64-12.1_2.6.35-22_1.img

glance image-list
+--------------------------------------+-------------------+-------------+------------------+----------+--------+
| ID                                   | Name              | Disk Format | Container Format | Size     | Status |
+--------------------------------------+-------------------+-------------+------------------+----------+--------+
| 386652ae-a92f-48fc-b3ec-84850ebf4ae1 | tty-linux-kernel  | aki         | aki              | 4404752  | active |
| 3abc2f1f-ca96-490d-9e14-3cc5e10eb685 | tty-linux-ramdisk | ari         | ari              | 96629    | active |
| 83b2c958-1b8c-435b-8ae1-7c49e06c60a2 | tty-linux         | ami         | ami              | 25165824 | active |
+--------------------------------------+-------------------+-------------+------------------+----------+--------+
}

# KVM configuration

# verify hardware support
apt-get install cpu-checker
root@stratus:~# kvm-ok {
INFO: /dev/kvm does not exist
HINT:   sudo modprobe kvm_intel
INFO: Your CPU supports KVM extensions
KVM acceleration can be used}
# or check output of
egrep '(vmx|svm)' --color=always /proc/cpuinfo

Pre-configuring the network
ip link set eth0 promisc on

add to /etc/network/interfaces {
# Bridge network interface for VM networks
auto br100
iface br100 inet static
address 192.168.100.1
netmask 255.255.255.0
bridge_stp off
bridge_fd 0
}

apt-get install bridge-utils
sudo brctl addbr br100
/etc/init.d/networking restart

# create mysql database
mysql -u root -p {
mysql> CREATE DATABASE nova;
mysql> GRANT ALL ON nova.* TO 'nova'@'%' IDENTIFIED BY '****';
mysql> GRANT ALL ON nova.* TO 'nova'@'localhost' IDENTIFIED BY '****';
mysql> FLUSH PRIVILEGES;
}

# install nova
sudo apt-get install nova-compute nova-volume nova-novncproxy novnc nova-api nova-ajax-console-proxy nova-cert nova-consoleauth nova-doc  \
nova-scheduler nova-network

backup conf files
cd /etc/nova/
cp -p api-paste.ini api-paste.ini.orig
diff api-paste.ini.orig api-paste.ini {
124,126c124,126
< admin_tenant_name = %SERVICE_TENANT_NAME%
< admin_user = %SERVICE_USER%
< admin_password = %SERVICE_PASSWORD%
---
> admin_tenant_name = service
> admin_user = nova
> admin_password = *****
}

replace nova.conf {
}

for service in nova-api nova-compute nova-network nova-scheduler nova-novncproxy nova-volume nova-cert nova-consoleauth
do
service $service stop
done
nova-manage db sync

for service in nova-api nova-compute nova-network nova-scheduler nova-novncproxy nova-volume nova-cert nova-consoleauth
do
service $service start
done
service libvirt-bin restart
/etc/init.d/rabbitmq-server restart

vgcreate nova-volumes /dev/sda4
service nova-volume restart
nova-manage network create private --fixed_range_v4=192.168.100.0/24 --bridge_interface=br100 --num_networks=1 --network_size=256

# as root:
nova-manage service list
}

### registre vm images ###
{
mkdir stackimages
wget -c https://launchpad.net/cirros/trunk/0.3.0/+download/cirros-0.3.0-x86_64-disk.img -O stackimages/cirros.img
glance image-create --name=cirros-0.3.0-x86_64 --disk-format=qcow2 --container-format=bare < stackimages/cirros.img
}

### running VM images ###

nova secgroup-list
nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0

If you cannot start VMs after installation without rebooting, it's possible the permissions are not correct. This can happen if you load the KVM
```

```
module before you've installed nova-compute. To check the permissions, run ls -l /dev/kvm to see whether the group is set to kvm.
If not, run sudo udevadm trigger.

# set state to active and then delete an instance in error.
nova reset-state --active 1cd534f1-d0ea-4289-8723-87833989dfd9
nova delete 1cd534f1-d0ea-4289-8723-87833989dfd9

missing

### Installing Dashboard ###

apt-get install -y memcached libapache2-mod-wsgi openstack-dashboard
ufw allow https/tcp
ufw allow http/tcp
# for vnc console access
ufw allow 6080/tcp
ufw enable

a2enmod
ln -sf /etc/apache2/sites-available/default-ssl /etc/apache2/sites-enabled/
# add
RedirectMatch ^/$ https://stratus.at.mt.ut.ee/horizon
# to both
/etc/apache2/sites-available/default
/etc/apache2/sites-available/default-ssl
service apache2 restart
```

Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, Valdur Kadakas (date of birth: 01/01/1978), herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

Establishing Scientific Computing Clouds on Limited Resources using OpenStack
supervised by Pelle Jakovits and Satish Narayana Srirama,

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 14.05.2013