

# **SETTING UP THE OPENSTACK WITH XEN HYPERVISOR & DEVELOPMENT OF A SIMULATOR ON A MULTI-TIER SYSTEM**

*A Project Report*

*Submitted by*

**SHUBHAM AGRAWAL**

**EE10B096**

*In partial fulfilment of the requirements  
for the award of the degree of*

**Master of Technology**

**in**

**Electrical Engineering**



**DEPARTMENT OF ELECTRICAL ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY MADRAS**

**2014-2015**

## THESIS CERTIFICATE

This is to certify that the thesis titled **SETTING UP THE OPENSTACK WITH XEN HYPERVISOR & DEVELOPMENT OF A SIMULATOR ON A MULTI-TIER SYSTEM** submitted by **Shubham Agrawal [EE10B096]**, to the Indian Institute of Technology Madras, Chennai for the award of the degree of **Master of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

**Dr. Ramakrishna Pasumarthi**

Project Guide

Assistant Professor

Dept. of Electrical Engineering

IIT-Madras, 600 036

Place: Chennai

Date: 24<sup>th</sup> May 2015

## ACKNOWLEDGEMENTS

I would like to express my appreciation and gratitude to my advisor **Dr. Ramkrishna Pasumarthi** for giving me the opportunity to work on this highly exciting and challenging project. Your invaluable guidance and motivation, throughout the project helped me to grow as a researcher.

The following research work for setting the OpenStack on the front end with the XEN hypervisor on the back end in a multi-tier environment & developing a simulator for the same would not have been possible without the continuous guide and support of **P.S. Sai Krishna**.

I also thank the member of **Cloud Computing** lab, **Mr. Manish Heda**, **Mr. Arejeet Nag** and **Ms. Niharika Challapalli** for their valuable help during the study. Lastly I would like to thank my family and friends for their constant motivation and encouragement.

## **ABSTRACT**

**KEYWORDS:** OpenStack, XEN Hypervisor, Xampp, Virtualization, Server Consolidation, Cloud, Multi-tier system, Simulator.

Cloud computing is the emerging paradigm of computing services. As Clouds are complex, large-scale, and heterogeneous distributed systems, management of their resources is a challenging task. In this work we have tried to set a multi-tier system wherein the front end is OpenStack, cloud computing software and the back end is Xen hypervisor. The paper talks about deploying an application on virtual machines in Xen environment by developing a simulator in java for the same with the aim of managing the resources from the OpenStack Platform. The simulator sits on the virtual machines and makes use of the Xampp server to use the data from MySQL Database server and deploying the application on Apache Web Server on cloud. The simulator was developed in Java language and also makes use of PHP script. This is a big untouched field where many researchers are trying to develop the best possible technology for utilization based multi-resource system. Elastic n-tier applications have non-stationary workloads that require adaptive control of resources allocated to them. This presents not only an opportunity in pay-as-you-use clouds, but also a challenge to dynamically allocate virtual machines appropriately.

**GENERAL TERMS:** Design, Experimentation, Management, Simulation.

# TABLE OF CONTENTS

<b>ACKNOWLEDGEMENTS</b>	<b>i</b>
<b>ABSTRACT</b>	<b>ii</b>
<b>LIST OF FIGURES</b>	<b>vi</b>
<b>ABBREVIATIONS</b>	<b>vii</b>
<b>1. INTRODUCTION</b>	<b>1</b>
<b>2. CLOUD COMPUTING – OVERVIEW AND OPENSTACK SETUP</b>	<b>4</b>
2.1. Overview and Characteristics . . . . .	4
2.2. Deployment Model Classification . . . . .	6
2.3. Service Model Classification . . . . .	6
2.4. OpenStack – Open Source Cloud Computing . . . . .	8
2.5. OpenStack Architecture . . . . .	8
2.6. Description of the Services . . . . .	10
2.7. OpenStack Framework Setup . . . . .	12
2.7.1 System Requirements . . . . .	12
2.7.2 Networking Mode . . . . .	14
2.7.3 Provisioning . . . . .	15
2.7.4 Installation . . . . .	18
2.8. Installing Web Server and MySQL on Compute Node . . . . .	24

<b>3.</b>	<b>VM SETUP IN XEN ENVIRONMENT</b>	<b>27</b>
3.1.	XEN Overview .....	27
3.2.	XEN Setup .....	28
3.2.1	OS Installation .....	28
3.2.2	Installing XEN .....	29
3.2.3	Network Configuration .....	29
3.2.4	Recommended Bridge Setting .....	30
3.2.5	Choice of Toolstacks .....	30
3.3.	Grub Setup to Boot XEN Hypervisor .....	31
3.4.	VMM Setup .....	32
3.5.	Setting up VMs .....	33
3.6.	Overview Diagram .....	34
<b>4.</b>	<b>A BRIEF ABOUT XAMPP AND HTTPERF</b>	<b>35</b>
4.1.	XAMPP Overview.....	35
4.2.	Configuring the XAMPP Environment.....	35
4.3.	Setting up MySQL Database .....	36
4.4.	HTTPERF Overview.....	37
4.4.1	Configuring Httpperf .....	37
4.4.2	Building Httpperf .....	38
4.4.3	Example .....	38
4.4.4	Selecting Timeout Values .....	39
<b>5.</b>	<b>DEVELOPMENT OF SIMULATOR IN JAVA ENVIRONMENT</b>	<b>41</b>
5.1.	Simulator Overview .....	41
5.2.	Simulator Setup .....	43
5.3.	Application – Differential Equation Solver .....	44
5.4.	Java Based Approach with MySQL Procedures .....	45
5.5.	Expected Results and Conclusion .....	46

5.6. Future Work. . . . .	46
<b>A. APPENDIX A – SAMPLE CODES</b>	<b>48</b>
A.1 Sample PHP Script . . . . .	48
A.2 Sample Procedure in MySQL . . . . .	49
A.3 Sample Java Script . . . . .	50
<b>B. APPENDIX B – REFERENCES</b>	<b>54</b>

## LIST OF FIGURES

2.1	Service Models .....	4
2.2	Example image to demonstrate Security Group .....	22
2.3	Example image to show launching of an image .....	23
2.4	Instances and Volumes .....	23
2.5	Host connection through terminal .....	24
3.1	Overview of the VM setup in Xen Environment .....	34
5.1	Experimental Setup for Simulator .....	43
5.2	Working of Simulator in 3-Tier Environment Simulator .....	42



## ABBREVIATIONS

WWW	World Wide Web
CPU	Central Processing Unit
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
QoS	Quality of Service
SLA	Service Level Agreement
SLO	Service Level Objectives
OS	Operating System
VM	Virtual Machine
VMM	Virtual Machine Manager
DB	Database
ODE	Ordinary Differential Equation
IO	Input Output
MIMO	Multiple Input Multiple Output
IP	Internet Protocol

# **CHAPTER 1**

## **INTRODUCTION**

Cloud computing found its origin in the success of server virtualization and the potential to run IT more efficiently through server consolidation. Visionaries came up with idea to bring virtualization to a next level by implementing early storage and network virtualization techniques that could be applied systematically across all the machines in a single data center.

Add to this self-provisioning and auto scaling, and cloud computing was born. At the time it was called utility computing, however, and only Amazon – an online bookstore, was using it as a way to manage their internal computing resources. Amazon saw a growth in popularity of its EC2 (compute) and S3 (storage) services. The Amazon API was being used by thousands of developers and many more customers to deploy and run infrastructure in the cloud.

Internet applications such as online news, retail, and financial sites have become common place in recent years. Modern Internet applications are complex software systems that employ a multi-tier architecture and are replicated or distributed on a cluster of servers. Each tier provides certain functionality to its preceding tier and makes use of the functionality provided by its successor to carry out its part of the overall request processing. For instance, a typical e-commerce application consists of three tiers—a front-end Web tier that is responsible for HTTP processing, a middle tier Java enterprise server that implements core application functionality, and a backend database that stores product catalogs and user orders. In this example, incoming requests undergo HTTP processing, processing by Java application server, and trigger queries or transactions at the database.

Cloud computing has evolved through a number of phases which include grid and utility computing, application service provision (ASP), and Software as a Service (SaaS). It's still at an early stage, with a motley crew of providers delivering a slew of cloud-based services, from full-blown application to storage services. Characteristics of cloud

like autonomous and dynamic provisioning, scalability, optimization benefits, networked access and metered servicing have propelled this technology to the forefront Gong *et al.* (2010). Multiple users were capable of accessing a central computer through dumb terminals, whose only function was to provide access to the mainframe. Because of the costs to buy and maintain mainframe computers, it was not practical for an organization to buy and maintain one for every employee. Nor did the typical user need the large (at the time) storage capacity and processing power that a mainframe provided. Providing shared access to a single resource was the solution that made economical-sense for this sophisticated piece of technology.

Among the three standard models of deployment of Cloud Gibson *et al.* (2012), SaaS is the most popular and widely used model of cloud computing. SaaS uses the Internet to deploy applications on the Cloud that are managed by third party vendors He (2010). Users access the application via the web browser on their side. This eliminates the need to download and install software to run these applications. SaaS enables users to use applications without being responsible for maintaining the data, O/S, virtualization, servers, storage and networking. In the SaaS model of deployment, the user is expected to only give the inputs required for the particular applications to get back the outputs without being responsible for computer administration, software installation or system details. System details include properties like the number of nodes to be used, the amount of storage needed and the operating system deployed etc.

The contribution of this work is to be able to develop a simulator which has an application which solves a differential equation wherein it takes the required coefficients and constants from the database server and update the time-bound results back on the server. The application can be run on the web server on the front end, in a cloud environment OpenStack and request can be made to VMs sitting on the back end Xampp server. Several requests can come in simultaneously using httpperf and is communicated over the network. The code to fetch the query from the database server and to solve the differential equation has been developed in Java language and PHP scripting language has been used to display the results on the browser.

The overview of the cloud computing and setting up of OpenStack cloud environment has been discussed in a great detail in chapter 2. Chapter 3 talks about the setup and configuration of virtual machines as per the experimental requirement in the XEN environment which has been setup on the backend machine. Chapter 4 shades light on the Xampp platform and talks about its setup and usage. It also discuss about running the instance of httpperf to generate multiple request to the server. An example request has been included to have a better understanding of the working and results. Chapter 5 looks into the experiment and results. The experimental part includes deploying an application written in java which uses SQL queries to connect to backend database server and fetch the queries to run the java code. The future scope in this attempt has also been discussed to give a better understanding of the results we are trying to achieve. Finally the sample codes has been included in Appendix A to make the understanding clearer of the PHP connection to server to fetch the queries and writing the procedures in SQL and java application to solve the differential equation using the Runge Kutta method.

## **CHAPTER 2**

### **Cloud Computing – Overview and OpenStack Setup**

#### **2.1. Overview and Characteristics**

Over the last few years clouds have become the buzzword in computing. But ask someone what cloud computing is and they're likely to give you a very different answer to the person standing next to them. Cloud computing covers anything that involves delivering hosted services over the internet.

Cloud computing involves application systems which are executed within the cloud and operated through internet enabled devices. Cloud computing relies on sharing of resources to achieve coherence and economies of scale, similar to a utility (like the electricity grid) over a network. At the foundation of cloud computing is the broader concept of converged infrastructure and shared services. Cloud computing, or in simpler shorthand just "the cloud", focuses on maximizing the effectiveness of the shared resources. Cloud resources are usually not only shared by multiple users but are also dynamically reallocated per demand.

The characteristics of cloud computing are

- **Self-service Provisioning:** The cloud allows users to deploy their own sets of computing resources (machines, network, storage, etc.) as needed without the delays and complications typically involved in resource acquisition.
- **Dynamic Provisioning:** The provisioning of resources can be done as per the SLA.
- **Elasticity and Scalability:** Unlike the individual users for whom the usage is typically fluctuating, a cloud can easily accommodate rapid increase or

decrease in resource demand. Thus the user can utilize as per the usage and avoid the cost of idle infrastructure.

- **Network Access and Storage Virtualization:** Cloud services can be accessed from anywhere via the Internet and from any type of device. They also provide storage capability independent of device and location.
- **Metered Services:** The usage of resources is metered and the customers are billed accordingly.

Clouds enable access to leased computing power and storage capacity from your desktop. Clouds are a proprietary technology. Only the resource provider knows exactly how their cloud manages data, job queues, and security requirements and so on. Small to medium to large commercial businesses or researchers with generic IT needs gets benefited from the cloud. Some of the benefits are

- **Flexibility:** Users can quickly outsource peaks of activity without long term commitment
- **Reliability:** Provider has financial incentive to guarantee service availability (Amazon, for example, can provide user rebates if availability drops below 99.9%)
- **Ease of use:** Relatively quick and easy for non-expert users to get started but setting up sophisticated virtual machines to support complex applications is more difficult.
- **Optimization Benefits:** The cloud can maximize the usage and increase the efficiency of existing infrastructure resources. This can help to reduce capital expenditure and extend infrastructure lifecycle.

Talking about some of the disadvantages of using cloud services, here are some of the drawbacks

- **Generality:** Clouds do not offer many of the specific high-level services currently provided by grid technology.

- Security: Users with sensitive data may be reluctant to entrust it to external providers or to providers outside their borders.
- Opacity: The technologies used to guarantee reliability and safety of cloud operations are not made public.
- Rigidity: The cloud is generally located at a single site, which increases risk of complete cloud failure.
- Provider lock-in: There's a risk of being locked in to services provided by a very small group of suppliers.

## **2.2. Deployment Model Classification**

Depending on the deployment models, clouds are classified as

1. Public Cloud: A public cloud can be accessed by any user with an internet connection and is intended for public use.
2. Private Cloud: A private cloud is established and operated solely by a specific group or organization and access is restricted to that group.
3. Community Cloud: A community cloud is shared among several organizations with community concerns and similar cloud requirements.
4. Hybrid Cloud: A hybrid cloud is a combination of several clouds, where the clouds are a mixture of public, private or community cloud.

## **2.3. Service Model Classification**

Depending on the service models, clouds are classified as

1. Software as a Service (SaaS): In this model the user purchases the ability to use a software application or service on the cloud. Eg: Google Docs

2. Platform as a Service (PaaS): In this model the user purchase access to platforms, enabling them to deploy their own application on the cloud. Eg: Google App Engine
3. Infrastructure as a Service (IaaS): In this model, the user is delivered infrastructure, namely servers, networks and storage. The user can deploy several Virtual Machines and run specific Operating Systems on them. Eg: Amazon EC2, Windows Azure etc.

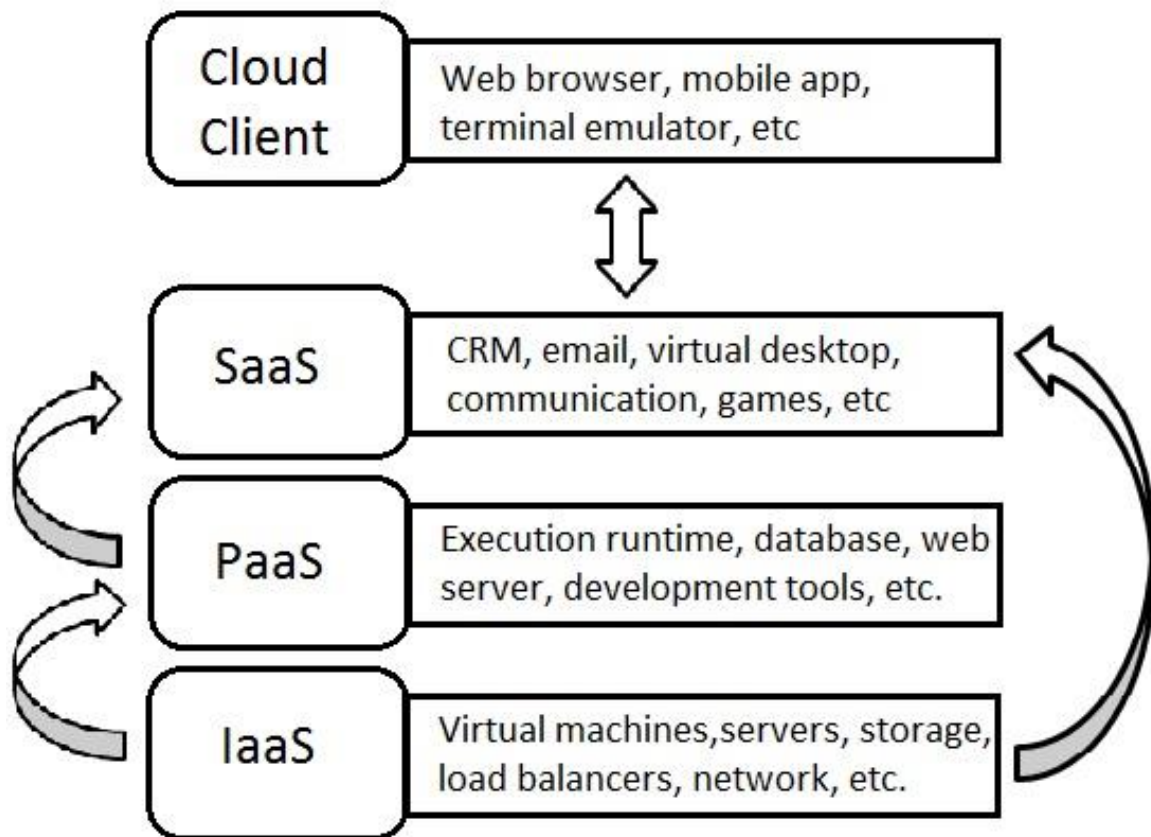


Figure 2.1: Service Models



## **2.4. OpenStack – Open Source Cloud Computing**

OpenStack is a set of software tools for building and managing cloud computing platforms for public and private clouds. OpenStack lets users deploy virtual machines and other instances which handle different tasks for managing a cloud environment on the fly. It makes horizontal scaling easy, which means that tasks which benefit from running concurrently can easily serve more or less users on the fly by just spinning up more instances.

OpenStack is a free and open-source cloud computing software platform. The technology consists of a series of interrelated projects that control pools of processing, storage, and networking resources throughout a data center which users manage through a web-based dashboard, command-line tools, or a RESTful API.

Users primarily deploy OpenStack as an infrastructure as a service (IaaS) solution. Providing infrastructure means that OpenStack makes it easy for users to quickly add new instance, upon which other cloud components can run. Typically, the infrastructure then runs a "platform" upon which a developer can create software applications which are delivered to the end users.

## **2.5. OpenStack Architecture**

OpenStack services that make up the OpenStack architecture are: Horizon, Nova, Neutron, Swift, Cinder, Keystone, Glance, Ceilometer and Heat. Swift and Cinder falls in Storage services, Keystone, Glance and Ceilometer comes under Shared services and Heat is a Higher-level service.

To design, install, and configure a cloud, cloud administrators must understand the logical architecture. OpenStack modules are one of the following types:

- **Daemon:** Runs as a daemon. On Linux platforms, a daemon is usually installed as a service.

- Script: Installs and tests of a virtual environment. For example, the `run_tests.sh` script installs and optionally tests a virtual environment for a service.
- Command-line interface (CLI): Enables users to submit API calls to OpenStack services through easy-to-use commands.

Basic architecture with legacy networking:

- The controller node runs the Identity Service, Image Service, dashboard, and management portion of Compute. It also contains the associated API services, MySQL databases, and messaging system.
- The compute node runs the hypervisor portion of Compute, which operates tenant virtual machines. By default, Compute uses KVM as the hypervisor. Compute also provisions and operates tenant networks and implements security groups. We can run more than one compute node.

Basic architecture with OpenStack Networking (Neutron):

- The controller node runs the Identity Service, Image Service, dashboard, and management portions of Compute and Networking. It also contains the associated API services, MySQL databases, and messaging system.
- The network node runs the Networking plug-in agent and several layer 3 agents that provision tenant networks and provide services to them, including routing, NAT, and DHCP. It also handles external (internet) connectivity for tenant virtual machines.
- The compute node runs the hypervisor portion of Compute, which operates tenant virtual machines. By default, Compute uses KVM as the hypervisor. The compute node also runs the Networking plug-in agent, which operates tenant networks and implements security groups. You can run more than one compute node.

## **2.6. Description of the Services**

### **Horizon**

Horizon is the dashboard behind OpenStack. It is the only graphical interface to OpenStack, so for users wanting to give OpenStack a try, this may be the first component they actually “see.” Developers can access all of the components of OpenStack individually through an application programming interface (API), but the dashboard provides system administrators a look at what is going on in the cloud, and to manage it as needed. In simpler words it provides a web-based self-service portal to interact with underlying OpenStack services, such as launching an instance, assigning IP addresses and configuring access controls.

### **Nova**

Nova is the primary computing engine behind OpenStack. It is a "fabric controller," which is used for deploying and managing large numbers of virtual machines and other instances to handle computing tasks. It manages the lifecycle of compute instances in an OpenStack environment. Responsibilities include spawning, scheduling and decommissioning of machines on demand.

### **Neutron**

Neutron provides the networking capability for OpenStack. It helps to ensure that each of the components of an OpenStack deployment can communicate with one another quickly and efficiently. Enables network connectivity as a service for other OpenStack services, such as OpenStack Compute. It provides an API for users to define networks and the attachments into them. Has a pluggable architecture that supports many popular networking vendors and technologies.

### **Swift**

Swift is a storage system for objects and files. Rather than the traditional idea of a referring to files by their location on a disk drive, developers can instead refer to a unique

identifier referring to the file or piece of information and let OpenStack decide where to store this information. This makes scaling easy, as developers don't have the worry about the capacity on a single system behind the software. It also allows the system, rather than the developer, to worry about how best to make sure that data is backed up in case of the failure of a machine or network connection. Stores and retrieves arbitrary unstructured data objects via a RESTful, HTTP based API. It is highly fault tolerant with its data replication and scale out architecture. Its implementation is not like a file server with mountable directories.

## **Cinder**

Cinder is a block storage component, which is more analogous to the traditional notion of a computer being able to access specific locations on a disk drive. This more traditional way of accessing files might be important in scenarios in which data access speed is the most important consideration. It provides persistent block storage to running instances. Its pluggable driver architecture facilitates the creation and management of block storage devices.

## **Keystone**

Keystone provides identity services for OpenStack. It is essentially a central list of all of the users of the OpenStack cloud, mapped against all of the services provided by the cloud which they have permission to use. It provides multiple means of access, meaning developers can easily map their existing user access methods against Keystone. It provides an authentication and authorization service for other OpenStack services. It also provides a catalog of endpoints for all OpenStack services.

## **Glance**

Glance provides image services to OpenStack. In this case, "images" refers to images (or virtual copies) of hard disks. Glance allows these images to be used as templates when deploying new virtual machine instances. It stores and retrieves virtual machine disk images. OpenStack Compute makes use of this during instance provisioning.

## **Ceilometer**

Ceilometer provides telemetry services, which allow the cloud to provide billing services to individual users of the cloud. It also keeps a verifiable count of each user's system usage of each of the various components of an OpenStack cloud. Think metering and usage reporting. It monitors and meters the OpenStack cloud for billing, benchmarking, scalability, and statistical purposes.

## **Heat**

Heat is the orchestration component of OpenStack, which allows developers to store the requirements of a cloud application in a file that defines what resources are necessary for that application. In this way, it helps to manage the infrastructure needed for a cloud service to run. It orchestrates multiple composite cloud applications by using either the native HOT template format or the AWS Cloud Formation template format, through both an OpenStack-native REST API and a Cloud Formation-compatible Query API.

## **2.7. OpenStack Framework Setup**

### **2.7.1 System Requirements**

#### **Compute Requirements**

1. **Physical Machines:** All OpenStack components must be installed on physical machines and not on virtual machines.
2. **Central Processing Units (CPUs):** It is recommended that each machine in your OpenStack cloud should contain either a Dual Core processor with a minimum of two, 2 GHz clock.
3. **Operating Systems:** OpenStack supports Ubuntu 12.04 LTS and some other Linux distributions.

4. Machine Clocks: Each OpenStack component machine and any other client machine clock must be synchronized (for example, using Network Time Protocol) at all the time, not just at installation.
5. Hypervisor: OpenStack Compute supports many hypervisors viz. XEN, KVM, QEMU, Hyper-V etc. For our work we have configured Xen hypervisor and have installed the nova-compute service in a para-virtualized VM.
6. Machine Access: Verify that all machines in your network allow root or sudo access and SSH login.

## **Storage and Memory Requirements**

The following are recommended:

- Each machine in your network needs a minimum of 30 GB of storage.
- Memory of 8 or 12 GB is recommended for the parent machine.
- Disk space is optimized for cost per GB.
- Each machine in your network should have at least 4 GB RAM or above for improved caching.

## **Network Requirements**

- One network interface card for external network traffic.
- Another card to communicate with other OpenStack nodes.
- All node controllers must have access to a minimum of 1 GB Ethernet Network connectivity.
- Depending on some configurations, OpenStack requires that you make available two sets of IP addresses. The first range is private, to be used only within the OpenStack system itself. The second range is public, to be routable to and from end-users and VM instances. Both sets must be unique to OpenStack, not in use by other components or application within your network.

## 2.7.2 Networking Mode

The architecture with OpenStack Networking (neutron) requires one controller node, one network node, and at least one compute node. The controller node contains one network interface on the management network. The network node contains one network interface on the management network, one on the instance tunnels network, and one on the external network. The compute node contains one network interface on the management network and one on the instance tunnels network.

Example 2.1 `/etc/network/interfaces`

```
# Internal Network
auto eth0
iface eth0 inet static
address 192.168.0.10
netmask 255.255.255.0
# External Network
auto eth1
iface eth1 inet static
address 10.0.0.10
netmask 255.255.255.0
```

After you configure the network, restart the daemon for changes to take effect:

```
# service networking restart
```

Set the host name of each machine. Name the controller node `controller` and the first compute node `compute1`. The examples in this guide use these host names.

Use the *hostname* command to set the host name:

```
# hostname controller
```

To configure this host name to be available when the system reboots, you must specify it in the `/etc/hostname` file, which contains a single line with the host name.

Finally, ensure that each node can reach the other nodes by using host names. You must manually edit the `/etc/hosts` file on each system. For large-scale deployments, use DNS or a configuration management system like Puppet.

```
127.0.0.1 localhost
192.168.0.10 controller
192.168.0.11 compute1
```

### **2.7.3 Provisioning**

#### **Nova**

Nova is the guts of the Orchestration. Scheduling, messaging, API, compute, object-store etc. are vital. Essex Notes:

- Nova-api accepts and responds to end user compute and volume API calls. It supports OpenStack API, Amazon's EC2 API and a special Admin API (for privileged users to perform administrative actions). It also initiates most of the orchestration activities (such as running an instance) as well as enforces some policy (mostly quota checks). In the Essex release, nova-api has been modularized, allowing for implementers to run only specific APIs.
- The nova-compute process is primarily a worker daemon that creates and terminates virtual machine instances via hypervisor's APIs (XenAPI for XenServer/XCP, libvirt for KVM or QEMU, VMwareAPI for VMware, etc.). The process by which it does so is fairly complex but the basics are simple: accept actions from the queue and then perform a series of system commands (like launching a KVM instance) to carry them out while updating state in the database.
- Nova-volume manages the creation, attaching and detaching of persistent volumes to compute instances (similar functionality to Amazon's Elastic Block Storage). It can use volumes from a variety of providers such as iSCSI or Rados Block Device in Ceph.



- The nova-network worker daemon is very similar to nova-compute and nova-volume. It accepts networking tasks from the queue and then performs tasks to manipulate the network (such as setting up bridging interfaces or changing iptables rules).
- The nova-schedule process is conceptually the simplest piece of code in OpenStack Nova: take a virtual machine instance request from the queue and determines where it should run (specifically, which compute server host it should run on).
- The queue provides a central hub for passing messages between daemons. This is usually implemented with RabbitMQ today, but could be any AMPQ message queue (such as Apache Qpid).
- The SQL database stores most of the build-time and run-time state for a cloud infrastructure. This includes the instance types that are available for use, instances in use, networks available and projects. Theoretically, OpenStack Nova can support any database supported by SQL-Alchemy but the only databases currently being widely used are sqlite3 (only appropriate for test and development work), MySQL and PostgreSQL.

## **Hypervisor**

Below is a list of the supported hypervisors with links to a relevant web site for configuration and use:

- KVM – Kernel-based Virtual Machine. The virtual disk formats that it supports it inherit from QEMU since it uses a modified QEMU program to launch the virtual machine. The supported formats include raw images, the qcow2, and VMware formats.
- LXC – Linux Containers (through libvirt), use to run Linux-based virtual machines.
- QEMU – Quick EMUlator, generally only used for development purposes.

- UML – User Mode Linux, generally only used for development purposes.
- VMWare ESX/ESXi 4.1 updates 1 runs VMWare-based Linux and Windows images through a connection with the ESX server.
- Xen – XenServer 5.5, Xen Cloud Platform (XCP), use to run Linux or Windows virtual machines. You must install the nova-compute service on DomU.

## Images

The script for this install is Ubuntu Precise Pangolin 12.04 from ubuntu.com. The default repo is:

```
http://docs.openstack.org/trunk/openstack-compute/admin/content/starting-images.html
```

The images use the EC2 repository for distributions and it seems to get out of sync. I switch to archive.ubuntu.com and will cover it in the screencast and steps in the post by editing the /etc/repo.

## Storage

For this demo we are just using the default local filesystem for storage. Swift appears to be lined up to replace this module from the Nova core.

- OpenStack Object Storage – OpenStack Object Storage is the highly-available object storage project in OpenStack.
- Filesystem – The default backend that OpenStack Image Service uses to store virtual machine images is the filesystem backend. This simple backend writes image files to the local filesystem.
- S3 – This backend allows OpenStack Image Service to store virtual machine images in Amazon's S3 service.

- HTTP – OpenStack Image Service can read virtual machine images that are available via HTTP somewhere on the Internet. This store is read only.

## Network

Nova-Network is the current module for networking in the Essex build for Layer2 bridging today. The current networking components are probably the least mature of the modules, since it is relying on the bridging kernel module in Linux.

Demo interfaces:

- Eth0 – NAT under VirtualBox. Leave it DHCP.
- Eth1 – Setup as a Host only. Needs to be done globally in VirtualBox the first time before you can add it to an instance.
- Eth2 – Compute nodes live here. Vlan100 will be bound to this interface from the install script and it will appear as br100 from ifconfig.

Nova Network commands and default logs:

`cat /var/lib/nova/networks/nova-br100.conf` – bridging config

`/var/log/nova/nova-network.log` – logs

`cat /etc/init/nova-network.conf` – config load

`/etc/init.d/nova-network` – upstart / startup config

Currently, Nova supports three kinds of networks, implemented in three “Network Manager” types respectively: Flat Network Manager, Flat DHCP Network Manager, and VLAN Network Manager.

### 2.7.4 Installation

Install VirtualBox. You will have three interfaces as listed above. Eth0 will be left NATing in VirtualBox, Eth1 (virbr0 in Vbox) is host only 172.16.0.0/16 and Eth2 (virbr1 in Vbox) is 10.0.0.0/8.

We are using Ubuntu 12.04 LTS (Precise Pangolin) daily build.

Once booted optionally update your OS and su root password

```
#sudo passwd root
#su
#apt-get update; apt-get upgrade
```

Installing the guest addons found in the VirtualBox 'Devices' menu is also recommended. This will optimize perf depending on the box you are doing this on.

I tend to gut networking managers on a fresh install; you could just stop the service if you would rather.

```
#apt-get purge network-manager
```

Edit your networking to reflect what is below.

```
#nano /etc/network/interfaces
```

(beginning)

#Primary interface NAT interface

auto eth0

iface eth0 inet dhcp

#public interface – The API village

auto eth1

iface eth1 inet static

address 172.16.0.1

netmask 255.255.0.0

network 172.16.0.0

broadcast 172.16.255.255

#Private Vlan Land of Compute Nodes

auto eth2

iface eth2 inet manual

up ifconfig eth2 up

(end)

Restart the networking service.

```
#/etc/init.d/networking restart
```

Test Inet connectivity. Ping google.com or some Inet host to make sure you have good connectivity.

Pull down the installer script. Uksysadmin did this script and it worked very well. There is a bunch out there at the moment most break quickly. This one works very well and beats compiling each component from scratch just for testing.

Install git from the repos:

```
#apt-get install git
```

Clone the installer:

```
#git clone
https://github.com/uksysadmin/OpenStackInstaller.git
#cd OpenStackInstaller
#git checkout essex
```

Run the installer, this will add the network interfaces, DHP and addresses you need automagically:

```
#./OSinstall.sh -F 172.16.1.0/24 -f 10.1.0.0/16 -s 512 -t
demo -v qemu
```

Edit nova.conf and add the 'my\_ip' value to the end of nova.conf:

```
#nano /etc/nova/nova.conf
#-my_ip=172.16.0.1
```

Wordpress mangles '-' There are two of the before 'my' e.g. "-- my\_ip"

You should see the following processes running at this point. If not start scouring logs or respawn with

```
/etc/init.d/ restart.
```

```
#ps -ea | grep glance
1345 ? 00:00:00 glance-api
1346 ? 00:00:00 glance-registry
```

```
# ps -ea | grep nova
1340 ? 00:00:21 nova-compute
1341 ? 00:00:16 nova-network
1342 ? 00:00:03 nova-api
1343 ? 00:00:10 nova-scheduler
1344 ? 00:00:00 nova-objectstor
```

In Ubuntu 12.04, the database tables are under version control, fix it with the following on a new install to prevent the Image service from breaking possible upgrades:

```
#glance-manage version_control 0
#glance-manage db_sync
```

Restart Glance just in case, probably not nessecary:

```
#!/etc/init.d/glance-api restart
#!/etc/init.d/glance-registry restart
```

Download the image. If you don't see "saved [197289496/197289496]" and see "saved []" with an empty vaue the image will not show in the Nova dashboard.

```
#./upload_ubuntu.sh -a admin -p openstack -t demo -C
172.16.0.1
```

If you need to have a couple of attempts you may need to delete the image download in /tmp/\_\_upload/

# rm /tmp/\_\_upload/\* (\*\*only if you get ubuntu 12.04 x64 now available in Glance (empty) or nothing in your images menu in Dashboard later in the install\*\*).

Saving to: /tmp/\_\_upload/ubuntu-12.04-server-cloudimg-x64.tar.gz'

```
100%[=====>]          197,289,496
610K/s   in 5m 31s

2012-04-20  01:49:56  (581  KB/s)  -  /tmp/__upload/ubuntu-
11.10-server-cloudimg-i386.tar.gz'          saved
[197289496/197289496]
```

Ubuntu 11.10 i386 now available in Glance (d19924ae-56e7-41bc-9f02-bbc8d7265cdf)

Log into the dashboard at <http://172.16.0.1>

Username: demo

Password: openstack

Make sure you see “Ubuntu 12.04 x64 Server” under Project -> Images and Snapshot. If you don’t troubleshoot the image upload and Glance install/config.

Under projects –Demo – Go to Access & Security

And create a keypair and save the key somewhere you can get to it later.

Edit the default rules with:

IP Protocol From Port To Port Source

TCP 22 22 0.0.0.0/0 (CIDR)

ICMP -1 -1 0.0.0.0/0 (CIDR)

TCP 8080 8080 0.0.0.0/0 (CIDR)

TCP 80 80 0.0.0.0/0 (CIDR)

The screenshot shows a web interface titled "Edit Security Group Rules". It contains a table of "Security Group Rules" with columns: IP Protocol, From Port, To Port, Source, and Actions. There are four rules listed: TCP 22, TCP 8080, TCP 80, and ICMP -1. Each rule has a "Delete Rule" button. Below the table is an "Add Rule" section with fields for IP Protocol (dropdown), From Port, To Port, Source Group (dropdown), and CIDR (text input). The "Add Rule" button is at the bottom right.

	IP Protocol	From Port	To Port	Source	Actions
<input type="checkbox"/>	TCP	22	22	0.0.0.0/0 (CIDR)	Delete Rule
<input type="checkbox"/>	TCP	8080	8080	0.0.0.0/0 (CIDR)	Delete Rule
<input type="checkbox"/>	TCP	80	80	0.0.0.0/0 (CIDR)	Delete Rule
<input type="checkbox"/>	ICMP	-1	-1	0.0.0.0/0 (CIDR)	Delete Rule

Displaying 4 items

**Add Rule**

IP Protocol:  From Port:  To Port:  Source Group:  CIDR:

Figure 2.2: Example image to demonstrate Security Group Rules

## Launching an Image and Using Instances

Under project -> Images & Snapshot -> click launch -> name the instance, choose the default security policy. Also, choose the keypair you created from the dropdown.

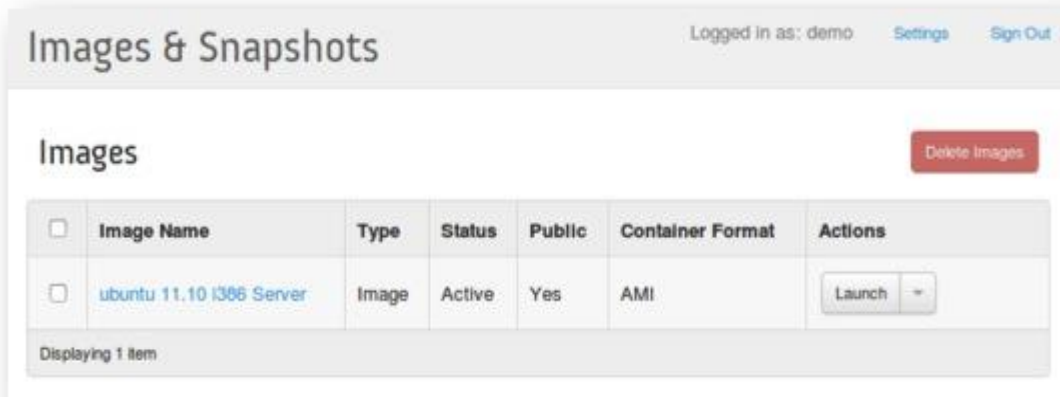


Figure 2.3: Example image to show launching of an image

After you launch your instances you should see them provisioned and available in Dashboard.



Figure 2.4: Instances and Volumes

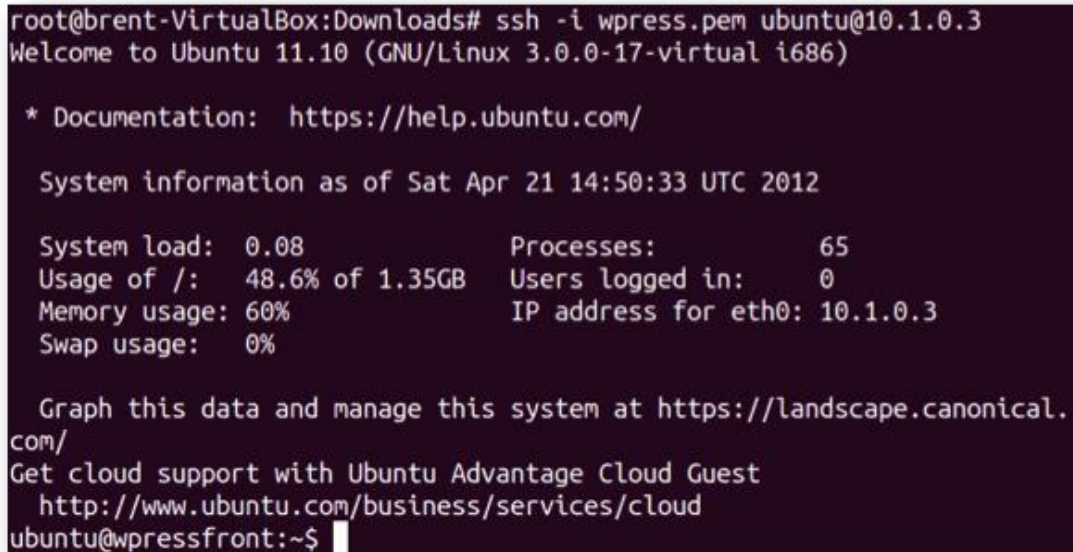
If that all looks like above then lets connect to the hosts using the key we saved earlier (get the IP from the dashboard under Instances & Volumes). Use Ubuntu as the login ID.



You can get the IP from the Dashboard or dump arp tables from the new node that was spun up.

cd into the directory with your key.

```
#ssh -i demo.pem ubuntu@10.1.0.x
```



```
root@brent-VirtualBox:Downloads# ssh -i wpress.pem ubuntu@10.1.0.3
Welcome to Ubuntu 11.10 (GNU/Linux 3.0.0-17-virtual i686)

* Documentation:  https://help.ubuntu.com/

System information as of Sat Apr 21 14:50:33 UTC 2012

System load:  0.08               Processes:            65
Usage of /:   48.6% of 1.35GB     Users logged in:     0
Memory usage: 60%               IP address for eth0: 10.1.0.3
Swap usage:   0%

Graph this data and manage this system at https://landscape.canonical.com/
Get cloud support with Ubuntu Advantage Cloud Guest
http://www.ubuntu.com/business/services/cloud
ubuntu@wpressfront:~$
```

Figure 2.5: Host connection through terminal

Now we have a self-provisioned abstracted compute node!

## 2.8. Installing Web Server and MySQL on Compute Node

We have OpenStack running on a couple of nodes now. Let's install WordPress on two nodes with web server running Apache on one node and MySQL on another compute node.

Apache2 web server install on the frontend node named wpressfront

MySQL install on the backend node named wpressback

Connect to a compute node:

```
#ssh -i wordp.pem ubuntu@10.1.0.5
```

(Find the address of the compute node from the Nova Dashboard)

```
#sudo passwd root
```

```
#su
```

## Install

```
#apt-get install libdbd-mysql-perl
```

```
#apt-get install mysql-client-core-5.1 mysql-client-5.1
```

```
#apt-get install mysql-server
```

## Edit MySql conf file

```
#nano /etc/mysql/my.cnf
```

# Instead of skip-networking the default is now to listen only on # localhost which is more compatible and is not less secure.

bind-address = x.x.x.x (or) 0.0.0.0 for all IPs on the host.

Log into MySQL on wpressback:

```
mysql -u root -p (Then prompted for passwd)
```

wpressdb = DB name

wpress = username

123 = password

```
mysql> CREATE DATABASE wpressdb; Query OK, 1 row affected  
(0.01 sec)
```

```
mysql> CREATE USER wpress; Query OK, 0 rows affected (0.00  
sec)
```

```
mysql> SET PASSWORD FOR 'wpress' = PASSWORD('123'); Query  
OK, 0 rows affected (0.00 sec)
```

```
mysql> GRANT ALL PRIVILEGES ON wpressdb.* TO 'wpress'  
IDENTIFIED BY '123'; Query OK, 0 rows affected (0.01 sec)  
mysql> exit
```

Now let's connect to the other node wpressfront and setup the Apache web server.

Connect to the second compute node and repeat the initial piece:

```
#ssh -i wordp.pem ubuntu@10.1.0.6 (find the address of the  
compute node from the Nova Dashboard)
```

```
#sudo passwd root
```

```
#su
```

```
#apt-get install apache2 libapache2-mod-php5 php5-mysql
```

Download first the latest WordPress version using this command from the terminal:

```
#wget http://wordpress.org/latest.tar.gz
```

Extract the archive file with this command:

```
#tar -xzvf latest.tar.gz
```

Create now a folder for WordPress in the /var/www directory using this command:

```
#mkdir /var/www/wordpress
```

Now move all WordPress files to this created folder using this command:

```
#cp -r wordpress/* /var/www/wordpress
```

If you are installing WordPress in a subdirectory not in the root directory (/var/www), then edit the apache2.conf file with this command:

```
#nano /etc/apache2/apache2.conf
```

At the end of the apache2.conf file, insert this line:

```
AddType application/x-httpd-php .html
```

I added 'chmod 777 /var/www/wordpress/' for the permissions.

```
chmod 777 /var/www/wordpress/
```

Then go to `http://(wpressfront IP)/wordpress/wp-admin/` or `http://(wpressfront IP)/wordpress/` and tinker around with the web app until it is how you want it.

## **CHAPTER 3**

### **VM Setup in XEN Environment**

#### **3.1. XEN Overview**

Xen is a hypervisor which uses a microkernel design, provides services that allow multiple computer operating systems to execute on the same computer hardware concurrently. The Xen Project hypervisor is an open-source type-1 or bare metal hypervisor, which makes it possible to run many instances of an operating system or indeed different operating systems in parallel on a single machine (or host). The Xen Project hypervisor is the only type-1 hypervisor that is available as open source. It is used as the basis for a number of different commercial and open source applications, such as: server virtualization, Infrastructure as a Service (IaaS), desktop virtualization, security applications, embedded and hardware appliances.

As of Ubuntu 11.10 (Oneiric), the default kernel included in Ubuntu can be used directly with the Xen hypervisor as the management (or control) domain (Dom0 or Domain0 in Xen terminology).

Xen runs in a more privileged CPU state than any other software on the machine. Responsibilities of the hypervisor include memory management and CPU scheduling of all virtual machines ("domains"), and for launching the most privileged domain ("dom0") - the only virtual machine which by default has direct access to hardware. From the dom0 the hypervisor can be managed and unprivileged domains ("domU") can be launched.

The dom0 domain is typically a version of Linux. User domains may either be traditional operating systems, such as Microsoft Windows under which privileged instructions are provided by hardware virtualization instructions or para-virtualized

operating system whereby the operating system is aware that it is running inside a virtual machine, and so makes hypercalls directly, rather than issuing privileged instructions.

Xen boots from a bootloader such as GNU GRUB, and then usually loads a paravirtualized host operating system into the host domain (dom0).

Here are some of the Xen Project hypervisor's key features:

- Small footprint and interface (is around 1MB in size). Because it uses a microkernel design, with a small memory footprint and limited interface to the guest, it is more robust and secure than other hypervisors.
- Operating system agnostic: Most installations run with Linux as the main control stack (aka "domain 0"). But a number of other operating systems can be used instead, including NetBSD and OpenSolaris.
- Driver Isolation: The Xen Project hypervisor has the capability to allow the main device driver for a system to run inside of a virtual machine. If the driver crashes, or is compromised, the VM containing the driver can be rebooted and the driver restarted without affecting the rest of the system.
- Paravirtualization: Fully paravirtualized guests have been optimized to run as a virtual machine. This allows the guests to run much faster than with hardware extensions (HVM). Additionally, the hypervisor can run on hardware that doesn't support virtualization extensions.

## **3.2. XEN Setup**

### **3.2.1. OS Installation**

During the install of Ubuntu for the partitioning method choose "Guided - use the entire disk and setup LVM". Then, when prompted to enter "Amount of volume group to use for guided partitioning:" Enter a value just large enough for the Xen Dom0 system, leaving the rest for virtual disks. Enter a value smaller than the size of your installation

drive. For example 10 GB or even 5 GB should be large enough for a minimal Xen Dom0 system. Entering a percentage of maximum size (e.g. 25%) is also a reasonable choice.

### 3.2.2. Installing XEN

Install a 64-bit hypervisor. (A 64-bit hypervisor works with a 32-bit dom0 kernel, but allows you to run 64-bit guests as well.)

```
# sudo apt-get install xen-hypervisor-amd64
```

As of Ubuntu 14.04, GRUB will automatically choose to boot Xen first if Xen is installed. If you're running a version of Ubuntu before 14.04, you'll have to modify GRUB to default booting to Xen.

Now reboot:

```
$ sudo reboot
```

And then verify that the installation has succeeded:

```
$ sudo xl list
```

Name	ID	Mem	VCPUs	State	Time(s)
Domain-0	0	945	1	r-----	11.3

### 3.2.3 Network Configuration

We will use the most common Xen network setup: bridged.

#### Disable Network Manager

If you are using Network Manager to control your internet connections, then you must first disable it as we will be manually configuring the connections.

```
# sudo stop network-manager
# echo "manual" | sudo tee /etc/init/network
manager.override
```

Using bridge-utils

```
# sudo apt-get install bridge-utils
```

In a bridged setup, it is required that we assign the IP address to the bridged interface.

Configure network interfaces so that they persist after reboot:

```
$ sudo vi /etc/network/interfaces
auto lo eth0 xenbr0
iface lo inet loopback

iface xenbr0 inet dhcp
    bridge_ports eth0

iface eth0 inet manual
```

Restart networking to enable xenbr0 bridge:

```
$ sudo ifdown eth0 && sudo ifup xenbr0 && sudo ifup eth0
```

The brctl command is useful for providing addition bridge information.

### 3.2.4 Recommended Bridge setting

For performance and security reasons, it's highly recommended<sup>2</sup> that netfilter is disabled on all bridges.

```
# sudo vi /etc/sysctl.conf

net.bridge.bridge-nf-call-ip6tables = 0
net.bridge.bridge-nf-call-iptables = 0
net.bridge.bridge-nf-call-arptables = 0

# sudo sysctl -p /etc/sysctl.conf
```

Note: These settings are created in /proc/sys/net. The bridge folder only appears to be created after first creating a bridge with the "brctl" command.

### 3.2.5 Choice of Toolstacks

One really great feature of the Xen Project hypervisor is that there are several choices of toolstacks that can be used to manage it. Before the introduction of libxenlight, there was a complicated and inefficient toolstack situation. The issue was that xend, xapi, libvirt,

etc. all needed to manage many common low-level operations, which led to code duplication, inefficiencies, bugs, and wasted effort (e.g fixing the same bug in all of the toolstacks). Further, the default toolstack at the time (xend) was difficult to understand, modify, and extend.

xl is a lightweight command line toolstack built using libxenlight. xl is shipped along with Xen and is the default toolstack as of Xen Project 4.1. xl was designed to be backwards compatible with the xm CLI for xend. It provides a straightforward command line interface to Xen's facilities for guest creation and management.

The Xen Project management API (xapi) is the default toolstack for XenServer. It is also used in some installations of CloudStack.

Libvirt is a virtualization API/toolkit used to manage various virtualization technologies. Originally designed to interface with xm, there is a libxenlight port of libvirt.

XEND is the previous toolstack and continues to be included as part of the Xen source releases. xl was designed to be command line compatible with the xm CLI to xend and this should provide the easiest upgrade path.

Note: Xend is the default toolstack in our machine as we are using a former version of Xen hypervisor.

### 3.3. Grub Setup to Boot Xen Hypervisor

Setup GRUB to boot the Xen Hypervisor

```
# sudo sed -i 's/GRUB_DEFAULT=.*\+/GRUB_DEFAULT="Xen 4.1-  
amd64"/' /etc/default/grub
```

Disable apparmor at boot

```
# sudo sed -i  
's/GRUB_CMDLINE_LINUX=.*\+/GRUB_CMDLINE_LINUX="apparmor=0"/  
' /etc/default/grub
```

Optional: Restrict "dom0" to 1GB of memory and 1 VCPU (example). (Only if you need to dedicate fix amount of memory for Xen "dom0").

```
# sudo gedit /etc/default/grub
```



After `GRUB_CMDLINE_LINUX="apparmor=0"` add the line  
`GRUB_CMDLINE_XEN="dom0_mem=1G,max:1G dom0_max_vcpus=1"`

Update Grub with the config changes we just made

```
# sudo update-grub
```

Reboot the server so that Xen boots on the server

```
# sudo reboot
```

Once the server is back online ensure that Xen is running

`cat /proc/xen/capabilities` should display `"control_d"`

Note: To stop or start `xcp-xapi`

```
# sudo /etc/init.d/xcp-xapi stop (or start)
```

Setup the default toolstack

```
# sudo gedit /etc/default/xen
```

-> Set `"TOOLSTACK=xapi"`

Disable `xend` from starting at boot

```
# sudo sed -i -e 's/xend_start$/#xend_start/' -e  
's/xend_stop$/#xend_stop/' /etc/init.d/xend
```

Note: Only `xend` the daemon needs to be disabled from starting, `"/etc/init.d/xend"` handles other things like modules and `xenfs`. Do not disable it from the runlevel. Disable service `xenddomains`.

```
# sudo update-rc.d xenddomains disable
```

Fix for `"qemu"` which emulates the console does not have the keymaps in the correct location

```
# sudo mkdir /usr/share/qemu; sudo ln -s /usr/share/qemu-  
linaro/keymaps /usr/share/qemu/keymaps
```

### 3.4. VMM Setup

Virtual Machine Manager or `virt-manager` is management software which manages virtual machines. The `virt-manager` application is a desktop user interface for managing virtual machines through `libvirt`. It primarily targets KVM VMs, but also manages Xen and LXC (linux containers). It presents a summary view of running domains, their live performance & resource utilization statistics. Wizards enable the creation of new domains, and configuration & adjustment of a domain's resource allocation & virtual

hardware. An embedded VNC and SPICE client viewer presents a full graphical console to the guest domain.

Installing it from the OS distribution

```
# apt-get install virt-manager
```

Note: Every time we start the Virtual machine manager we need to start the XEN Toolstack – Libvirt by using the command line.

```
# Xend start
```

Note: By default we have set the choice of toolstack in Xen to be Xend.

### 3.5. Setting up VMs

When you create a virtual machine from a blank virtual hard disk, you must configure the virtual machine so that you can install an operating system from a system CD, from an ISO image file in the library, or through a network service boot.

To create a virtual machine from a blank virtual hard disk:

- In the Actions pane in the Virtual Machine Manager Administrator Console, click New virtual machine.
- Complete the New Virtual Machine Wizard.

The wizard pane needs to be setup and the details for Selecting Source, VM Identity, Hardware Configuration, Destination Selection, VM Host selection, Path Selection, and additional properties has to be provided.

#### Installing an OS on the Virtual Machine

For installing an operating system from a system CD, from an ISO image file in the library, or through a network service boot, the VM must be configured first. For the first two methods, you must configure a virtual DVD drive to attach to a physical CD drive on the host or to the image file. For a network installation, you must configure a virtual network adapter.

We used an image file to install the operating system. For that the image file to the Virtual Machine Manager library must be added.

After loading the image file in new VM setup, we have to specify the memory, number of cores and RAM allocation for the new VM. A network setup need to be defined which is bridged-network in our case. Under Ethernet (MAC) address, specify a dynamic or static IP address for the virtual machine. With default rest of the setting, we install the OS (from image file) on the VM in a conventional manner. Once done, the VMs can be rebooted. For adding multiple virtual machines either we can clone the already existing virtual machines or follow the similar steps as above.

The virtual machine manager provides a time-saving option to ‘Clone’ the existing virtual machines. By cloning we can create multiple similar VMs with similar configurations. But the grub files needs to be configured in each of the new VM.

### 3.6. Overview Diagram

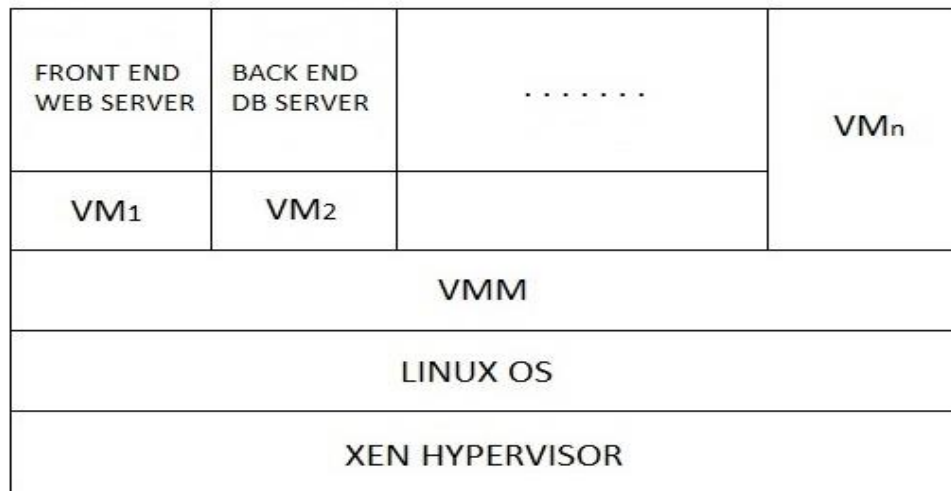


Figure 3.1: Overview of the VM setup in Xen Environment

## CHAPTER 4

### A Brief about Xampp and Httpperf

#### 4.1. Xampp Overview

XAMPP is a free and open source cross-platform web server solution stack package, consisting mainly of the Apache HTTP Server, MySQL database, and interpreters for scripts written in the PHP and Perl programming languages. The XAMPP open source package has been set up to be incredibly easy to install and to use. The name is acronym for X (meaning cross platform), Apache HTTP Server, MySQL, PHP and Perl.

The designers intended it for use only as a development tool, to allow website designers and programmers to test their work on their own computers without any access to the Internet. XAMPP also provides support for creating and manipulating databases in MySQL and SQLite among others. Once XAMPP is installed, it is possible to treat a localhost like a remote host by connecting using an FTP client.

It also comes with a number of other modules including OpenSSL and phpMyAdmin. Self-contained, multiple instances of XAMPP can exist on a single computer, and any given instance can be copied from one computer to another.

#### 4.2. Configuring the Xampp Environment

To install the Xampp, chose the flavor of the Linux OS and change the permission of the installer and then run it. In our case the Xampp server was installed and configured in all the VMs.

```
# chmod 755 xampp-linux-*-installer.run
# sudo ./xampp-linux-*-installer.run
```

To start or stop the Xampp,

```
# sudo /opt/lampp/lampp start (stop)
```

There is a graphical tool that you can use to manage your servers easily. You can start this tool with the following commands:

```
# cd /opt/lampp  
# sudo ./manager-linux.run (or manager-linux-x64.run)
```

To make request to the Web Server, start Apache in control panel, type `http://localhost` in your web browser. This would bring you a web page that lists XAMPP related details.

To put stuffs in Web folder, under XAMPP root directory there is a folder called `htdocs` which stores web site related stuff. For each web site you create, it's better to create a folder inside `htdocs` folder and then put content inside that to avoid conflicts. In our case the PHP code developed to fetch the query from MySQL database server and display it in the browser was put up in `htdocs` folder directly.

### **4.3. Setting up MySQL Database**

Xampp provides a number of modules, one of which is phpMyAdmin. The MySQL database server can be accessed and managed through it. The entire database can be created here. In our case multiple database were created and tables were added to it. The SQL procedures to fetch the query can be created and stored alongside the tables, which can be used in the java programs (stored in a remote machine) to fetch the values from the database. The setting up of database server in Xampp is straight forward and can be done through phpMyAdmin. The setting includes creating (or modifying an existing) database and adding tables to it and writing SQL queries as per need.

While creating a new database in phpMyAdmin SQL Database server, we need to give the permission for the root access to users. The entire Xampp server was setup in the virtual machines in our case. To access it from other networks, primarily from the frontend, we need to define the user access and set the password for it. The Internet protocol address needs to be defined at the time of setup to grant access to it over the network, without which it will fail to connect to the server. In the PHP and Java codes (for the experimental part), the same IP has to be used to set up a communication for a remote access. Let us see how to make multiple request to the server thorough Httpperf.

## 4.4. Httpperf overview

Httpperf is a tool for measuring web server performance. It provides a flexible facility for generating various HTTP workloads and for measuring server performance. The focus of httpperf is not on implementing one particular benchmark but on providing a robust, high-performance tool that facilitates the construction of both micro- and macro-level benchmarks. The three distinguishing characteristics of httpperf are its robustness, which includes the ability to generate and sustain server overload, support for the HTTP/1.1 and SSL protocols, and its extensibility to new workload generators and performance measurements.

### 4.4.1 Configuring Httpperf

Httpperf is using the standard GNU configuration mechanism. The following steps can be used to build it:

```
# mkdir build
# cd build
# SRCDIR/configure
# make
# make install
```

In this example, SRCDIR refers to the httpperf source directory. The last step may have to be executed as "root". To build httpperf with debug support turned on, invoke configure with option "--enable-debug".

By default, the httpperf binary is installed in /usr/local/bin/httpperf and the man-page is installed in /usr/local/man/man1/httpperf. You can change these defaults by passing appropriate options to the "configure" script. The configure script assumes that the OpenSSL header files and libraries can be found in standard locations (e.g., /usr/include and usr/lib). If the files are in a different place, you need to tell the configure script where to find them. This can be done by setting environment variables CPPFLAGS and LDFLAGS before invoking "configure". For example, if the SSL header files are installed in /usr/local/ssl/include and the SSL libraries are installed in /usr/local/ssl/lib, then the environment variables should be set like this:

```
CPPFLAGS="-I/usr/local/ssl/include"
```

```
LDLFLAGS="-L/usr/local/ssl/lib"
```

With these settings in place, "configure" can be invoked as usual and SSL should now be found. If SSL has been detected, the following three checks should be answered with "yes":

```
checking for main in -lcrypto... yes
```

```
checking for SSL_version in -lssl... yes
```

```
:
```

```
checking for openssl/ssl.h... yes
```

Note: you may have to delete "config.cache" to ensure that "configure" re-evaluates those checks after changing the settings of the environment variables.

#### **4.4.2 Building Httpperf**

It is crucial to run just one copy of httpperf per client machine. Httpperf sucks up all available CPU time on a machine. It is therefore important not to run any other (CPU-intensive) tasks on a client machine while httpperf is running. Httpperf is a CPU hog to ensure that it can generate the desired workload with good accuracy, so do not try to change this without fully understanding what the issues are.

#### **4.4.3 Example**

The simplest way to invoke httpperf is with a command line of the form:

```
# httpperf --server localhost --port 8080
```

This command results in httpperf attempting to make one request for URL `http://localhost:8080/`. After the reply is received, performance statistics will be printed and the client exits.

A more realistic test case might be to issue 1000 HTTP requests at a rate of 10 requests per second. This can be achieved by additionally specifying the `--num-conns` and `--rate` options. When specifying the `--rate` option, it's generally a good idea to also

specify a timeout value using the `--timeout` option. In the example below, a timeout of one second is specified.

```
# httpperf --server localhost --port 8080 --num-conns 100 --
rate 10 --timeout 1
```

The performance statistics printed by httpperf at the end of the test might look like this:

```
Total: connections 100 requests 100 replies 100 test-duration 9.905 s
Connection rate: 10.1 conn/s (99.1 ms/conn, <=1 concurrent connections)
Connection time [ms]: min 4.6 avg 5.6 max 19.9 median 4.5 stddev 2.0
Connection time [ms]: connect 1.4
Connection length [replies/conn]: 1.000
Request rate: 10.1 req/s (99.1 ms/req)
Request size [B]: 57.0
Reply rate [replies/s]: min 10.0 avg 10.0 max 10.0 stddev 0.0 (1 samples)
Reply time [ms]: response 4.1 transfer 0.0
Reply size [B]: header 219.0 content 204.0 footer 0.0 (total 423.0)
Reply status: 1xx=0 2xx=100 3xx=0 4xx=0 5xx=0
CPU time [s]: user 2.71 system 7.08 (user 27.4% system 71.5% total 98.8%)
Net I/O: 4.7 KB/s (0.0*10^6 bps)
Errors: total 0 client-timo 0 socket-timo 0 connrefused 0 connreset 0
Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0
```

There are six groups of statistics: overall results ("Total"), connection related results ("Connection"), results relating to the issuing of HTTP requests ("Request"), results relating to the replies received from the server ("Reply"), miscellaneous results relating to the CPU time and network bandwidth used, and, finally, a summary of errors encountered ("Errors").

#### 4.4.4 Selecting Timeout Values

Since the client machine has only a limited set of resource available, it cannot sustain arbitrarily high HTTP request rates. One limit is that there are only roughly 60,000 TCP port numbers that can be in use at any given time. Since it takes one minute for a TCP connection to be fully closed (leave the `TIME_WAIT` state), the maximum rate a client can sustain is about 1,000 requests per second.



The actual sustainable rate is typically lower than this because before running out of TCP ports, a client is likely to run out of file descriptors (one file descriptor is required per open TCP connection).

By default machine allows 1024 file descriptors per process. Without a watchdog timer, httpperf could potentially quickly use up all available file descriptors, at which point it could not induce any new load on the server (this would primarily happen when the server is overloaded). To avoid this problem, httpperf requires that the web server must respond within the time specified by option `--timeout`. If it does not respond within that time, the client considers the connection to be "dead" and closes it (and increases the "client-time" error count). The only exception to this rule is that after sending a request, httpperf allows the server to take some additional time before it starts responding (to accommodate HTTP requests that take a long time to complete on the server). This additional time is called the "server think time" and can be specified by option `--think-timeout`. By default, this additional think time is zero, so by default the server has to be able to respond within the time allowed by the `--timeout` option.

In practice, we found that with a `--timeout` value of 1 second, machine can sustain a rate of about 700 connections per second before it starts to run out of file descriptor (the exact rate depends, of course, on a number of factors). To achieve web server loads bigger than that, it is necessary to employ several independent machines, each running one copy of httpperf. A timeout of one second effectively means that "slow" connections will typically timeout before TCP even gets a chance to retransmit (the initial retransmission timeout is on the order of 3 seconds). This is usually OK, except that one should keep in mind that it has the effect of truncating the connection life time distribution.

## CHAPTER 5

### Development of Simulator in Java Environment

#### 5.1. Simulator Overview

Our primary objective here is to formulate a simulator which deploys applications in the VMs in the backend when run from the front end OpenStack platform. So we chose to solve a differential equation using the fourth order Runge-Kutta method as our application. The constants and coefficients required to solve the DEs are stored in MySQL Database in Xampp server in VM2 which serves as the backend machine. A java based approach has been adopted to do the purpose. Earlier we tried to do the same through the MATLAB script and C programming, but java turns out to be more flexible. A time bounded running of the application produces the solution to the DEs which is to be updated back to the database server and this serves the next starting point of the application. In simpler words, the next instance of application running makes use of the results of the previously-ran instance of the same application which serves as the new starting point. Also, each time the simulator communicates to the database server get the new coefficients and constants and to update the result. Next, the result thus generated is to be returned to the user in his web browser through the apache web server using the PHP script. The whole thing can be run as a single script or a bash script.

Thus a user sitting at the front end needs to run the simulator and the request goes to the backend server which does the job and returns the result to the user in his web browser on the frontend machine. On a scalability note, we planned to run thousand such scripts simultaneously using the requests generated through Httpperf. Thus our simulator works in a multi-tier environment where the frontend is OpenStack platform where the user is based and the backend is Xen environment where all our Virtual machines are sitting in which the job runs.

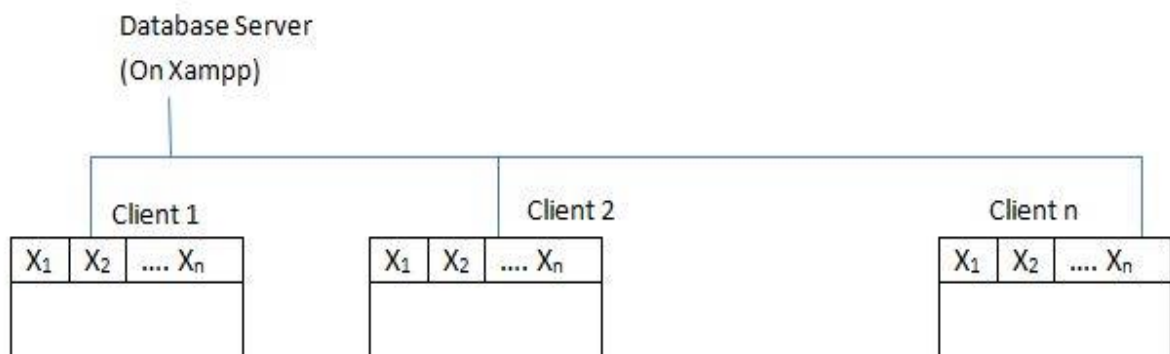
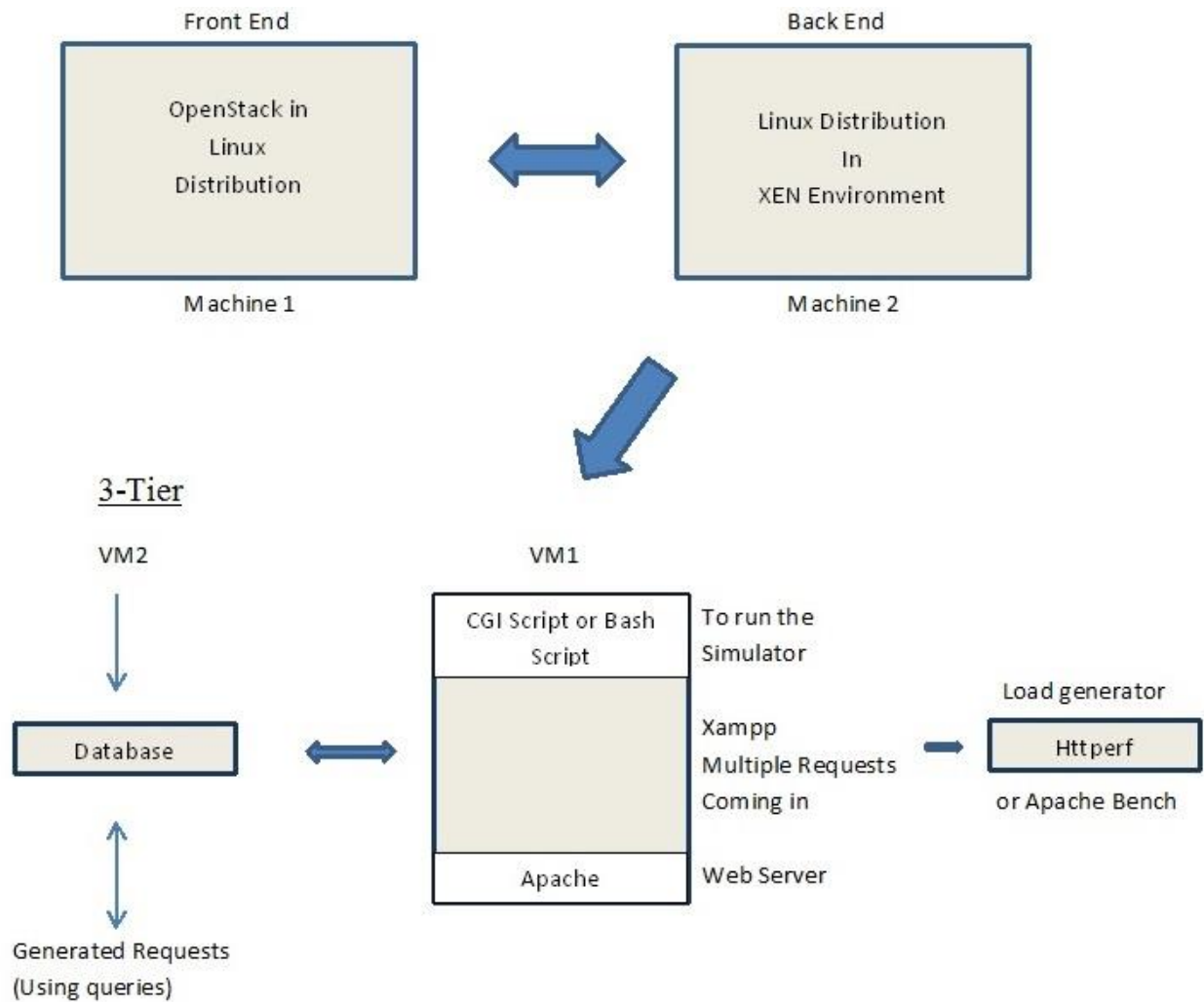


Fig 5.2: Working of the Simulator in 3-Tier Environment

The whole process makes use of networking between the different machines and the virtual machines and also the Xampp server sitting in VMs. Also it communicates both ways to the MySQL Database Server and Apache Web Server in the VMs. The Java and PHP script developed does the purpose. The Java script sets up a connection to the database server and simulates the application and the PHP script communicates to apache web server to produce the results.

## 5.2. Simulator Setup

In our multi-tier environment the frontend is OpenStack platform and the backend is Xen environment. The intra network has been set between the frontend and the backend machine for the communication. The backend machine in turn has Linux OS residing over XEN and the virtual machine manager sits inside the OS. Multiple virtual machines are installed in the VMM for various purposes. Everything is in Linux environment. The virtual machines communicate to the Xen using Xen-bridge adaptor. Further in the VMs Xampp has been setup and the servers are configured. In other words our Apache Web server and MySQL Database server resides in the VMs in the backend machine. However the results can be produced anywhere in any web browser. The simulator is to be made available over the network so as to run it from any machine and the results can be produced in the web browser of the same machine. To increase the scalability of the simulator, Httpperf can be used to generate hundreds of requests simultaneously.

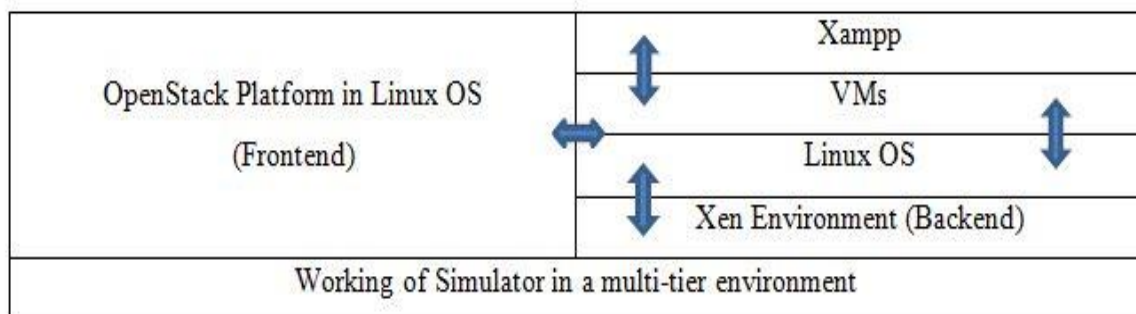


Figure 5.1: Experimental setup for simulator

The simulator code has been written in Java language. The code makes use of stored SQL procedures which has been written and stored along with the tables in database server. Procedure fetches the SQL query to the simulator. This is possible only after successfully setting up the connection with the database server which is again done in the same Java code. A sample code has been given in Appendix A. Earlier attempt to develop the simulator in C language and Matlab language were made but Java language proved to be much more flexible.

### **5.3. Application – Differential Equation Solver**

Differential equations, those that define how the value of one variable changes with respect to another, are used to model a wide range of physical processes. Our aim is to solve all type of DEs be it an initial value or two-point boundary problems. We will develop a class named ODESolver that will define a number of methods used to solve ODEs. Everything in java is defined within a class, so we need to define a class to encapsulate the DEs. With an initial value problem (IVP), values for all of the dependent variables are specified at the beginning of the range of integration. The solution is marched outward from the initial boundary by integrating the ODE at discrete steps of the independent variable. The dependent variables are computed at every step. IVP are simpler to solve because you only have to integrate the ODE one time. The solution of a two-point boundary value problem usually involves iterating between the values at the beginning and end of the range of integration. The most commonly used techniques to solve IVP ODEs are called Runge-Kutta schemes.

Runge-Kutta family of methods is a step wise integration algorithm which is most widely used algorithm. Starting from an initial condition, the ODE is solved at discrete steps over the desired integration range. Runge-Kutta techniques are robust and will give good results as long as very high accuracy is not required. Runge-Kutta methods are not the fastest ODE solver techniques but their efficiency can be markedly improved if adaptive step sizing is used. There are numerous Runge-Kutta schemes of various orders

of accuracy. The most commonly used scheme and the one we will implement in this chapter is the fourth-order Runge-Kutta algorithm.

#### **5.4. Java Based Approach with MySQL Procedures**

Java is a fairly recent Programming language. It became very popular with net-working, web design, and graphical user interfaces (GUIs). Java is an interpreted language which is run on a virtual machine. It is a very object-oriented language. Java is an interpreted language, and it involves run-time type identification (RTTI) and polymorphism. These properties give the programmer many advantages; however, this can make program execution slow for some applications. Java has become increasingly popular for many applications. However, it has not been used for many complex applications in scientific computing. Although there are a few packages in Java for the numerical methods, there have not yet been any major packages in java for the numerical solutions of ODEs.

The idea is to have an application running stored in a remote machine, which can be accessed from anywhere. So our application fetches the query required by it to run from the MySQL server sitting in the VMs of the remote machine, basically the cloud. To make this happen we have to first set the connection the database server in our application code which is written in java. Next, to fetch the queries we have to develop SQL Procedures which again can be called from the application code. The results are stored as a temporary variable in the application and then DESolver makes use of these variables to solve the DEs using Runge-Kutta algorithm. This whole code is developed in Java language. Next the results are again to be uploaded back to the Database servers. Again to do so, we have used another SQL Procedure which this time updates the Database with the results so that the same can be used the next time application is run. The same results along with the supporting graphs is then send to user and is displayed in his Web Browser using PHP script. The whole application is developed in Java language because of its robustness. Next time if we need to add extra component to the simulator, we just have to add a java class performing the desired function to the main class. The supporting packages and library files were easily available in java to make the simulator development easy.

## **5.5. Expected Results and Conclusion**

The working of simulator has been discussed in great details in earlier section. The expected end-task of the simulator is to upload the results back to the database server and return the same results along with the supporting graphs back to the user on the front-end or to the remote location of the user in his web browser. But there are still slight issues with its working as the simulator overall couldn't produce the results. Eventually, part by part working of the simulator was verified and the desired results are expected. So, finally we have a simulator cum software developed which is partially operational and the issues can be fixed with an advanced knowledge of working of Java and MySQL query operating.

The parts of the code developed for the simulator works fine individually and perform the desired role. But when put together as a bunch, it gives a runtime error in java. The connection establishment to the database server, calling of the MySQL Procedures, fetching the queries, solving the DEs, uploading the results back to the Database Server using another MySQL Procedures everything works fine individually. But when bundled together to work as a single fully-functional Simulator, it shows some issues. There are also some issues with the compiler as we are using the Java script for Runge-Kutta which is ten years old. And the modern day compiler which we have been using shows some compatibility issues. If these issues were solved, the simulator thus developed will be one of its kinds which use Java as its base language and also make use of apache and MySQL servers.

Note: To my knowledge no similar work has been done earlier in this regard and no other such implementation exists in the public domain. To my knowledge, there has never been such implementation in Java, a programming language that is very common and growing more popular every day.

## **5.6. Future Work**

The future work in this project includes solving the issues and going for some scientific application rather than solving Differential equations. Also, a better time bounded

algorithm can be designed for the application when hundreds of requests are being made to the simulator. Thus the scalability can be improved and the responsiveness of the application can be made better by using improved techniques. Also, system identification can be made for the whole multi-tier system governing the control theory. A control feedback system can be developed in case of unresponsiveness of server when the scalability gets enhanced. Utilization-based multi-resource controller can be formulated for the identified system for our case to enhance the performance. Thus, resource allocation to the server and the VMs can be made dynamically. Also, further work may include development of automated mechanism for saving the power consumption using virtualization.



## APPENDIX A

### SAMPLE CODES

#### A.1 Sample PHP Script

The PHP script below is just a sample code used for the purpose to make connections to the Database server (located in VMs) and fetch simple query from the tables and return the results in the Web browser. The query in the script could be modified as per the experimental requirements. Here the script fetches the result from TABLE\_3 from 'test' database from VM2.

```
<?php
$dbhost='192.168.122.82:3306';
$dbuser='vm2';
$dbpass='123';
$conn=mysql_connect($dbhost,$dbuser,$dbpass);
if(!$conn)
{
    die("could not connect".mysql_error());
}
echo "Connected";
mysql_select_db("test");
$sql="select * from TABLE_3";
$retval=mysql_query($sql,$conn);
if(!$retval)
{
    die("Could not get Data".mysql_error());
}
echo "<table border='1'>
<tr>
<th> Name </th>
<th> Std code </th>
<th> TelePhone </th>
<th> Mobile </th>
<th> Email </th>
<th> Name </th>
```

```

<th> Std code </th>
<th> TelePhone </th>
<th> Mobile </th>

</tr>";
while($row=mysql_fetch_array($retval,MYSQL_ASSOC))
{
echo "<tr>";
echo "<td>" . $row['Nodes'] . "</td>";
echo "<td>" . $row['RT'] . "</td>";
echo "<td>" . $row['Tput'] . "</td>";
echo "<td>" . $row['AB'] . "</td>";
echo "<td>" . $row['CD'] . "</td>";
echo "<td>" . $row['EF'] . "</td>";
echo "<td>" . $row['GH'] . "</td>";
echo "<td>" . $row['UI'] . "</td>";
echo "<td>" . $row['HY'] . "</td>";
}
mysql_close($conn);
?>

```

## A.2 Sample Procedure in MySQL

This procedure was written and configured in the ‘test’ database itself. The procedure gets the values from table\_1 by fetching the input query. This is just a sample procedure and can be modified as desired. The procedure later was used to develop the java code.

```

DROP PROCEDURE IF EXISTS showTable_1 $$ //MySQL returned
two column set (with 2 and 3 rows respectively)

```

```

CREATE PROCEDURE showTable_1 ()
BEGIN
SELECT * FROM Table_1;
END $$ //MySQL returned two column set (with 2 and 3 rows
respectively)

```

```

// the value of the Delimiter at the time of writing the
script should be changed to $$

```

```

// to run the procedure # showTable_1 () $$

```

### A.3 Sample Java Script

The java script can be used from the front end machine which makes a connection to the VM in the backend machine and calls the procedure to run the application (DE solver in this case). The connection with appropriate IP needs to be defined in the code. Again this is just a sample code used for our purpose and the modification can be done accordingly.

```
/*
 * TO RUN THIS CODE FIRST START THE XAMPP SERVER
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
/*
Example of the use of the class RungeKutta demonstrating
the use of instance methods in
solving a set of three ordinary differential equation
describing two consecutive first order processes:
y[0] --(k1)--> y[1] --(k2)--> y[2]
k1= rate constant for first process, k2 = rate constant for
second process
dy[0]/dt = -k1.y[0]
dy[1]/dt = k1.y[0] - k2.y[1]
dy[2]/dt = -k2.y[1]
*/

package desolver;

import java.sql.*;
import java.sql.DriverManager;

//import flanagan.integration.RungeKutta;
import desolver.package1.DerivnFunction;
import desolver.package1.DerivFunction;
import desolver.package1.IntegralFunction;
import desolver.package1.Integration;
import desolver.package1.RungeKutta;

/**
 *
 * @author vm2
 */
```

```

public class DEsolver {
/*
 * @param args the command line arguments
 */
public static void main(String[] args) {
// TODO code application logic here

try{
    Class.forName("com.mysql.jdbc.Driver");
    Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost/test", "
root","");
    CallableStatement cs;
    cs = conn.prepareCall("{call showTable_1}");
    ResultSet rest = cs.executeQuery();

    while (rest.next()) {
        System.out.println(rest.getString("AA") + "    "
+ rest.getString("BB") + "    " + rest.getString("CC") + "
" + rest.getString("DD") ); }

// Create instance of the class holding the derivative
evaluation method
    Derivn1 dn = new Derivn1();

// Assign values to constants in the derivative function
    double k1 = rest.getDouble("BB");
    double k2 = rest.getDouble("BB");
    double k1 = 2.0D;
    double k2 = 5.0D;
dn.setKvalues(k1, k2);

// Variable declarations
int i = 0;
int n = 3;           // number of differential equations
double h = 0.01      // step size
double x0 = 0.0D;     // initial value of x
double xn = 1.0D;     // final value of x
double[] y0 = new double[n]; // initial values of the y[i]
double[] yn = new double[n]; // returned value of the y[i]
at x = xn

```

```

// Assign initial values of y[i]
    y0[0] = rest.getDouble("CC");
    y0[1] = rest.getDouble("CC");
    y0[2] = rest.getDouble("CC");

// Create and instance of RungeKutta
    RungeKutta rk = new RungeKutta();

// Set values needed by fixed step size method
    rk.setInitialValueOfX(x0);
    rk.setFinalValueOfX(xn);
    rk.setInitialValuesOfY(y0);
    rk.setStepSize(h);

// Call Fourth Order Runge-Kutta method
    yn = rk.fourthOrder(dn);
// Output the results
System.out.println("Fourth order Runge-Kutta procedure");
System.out.println("The value of y[0] at x = " + xn + " is
" + yn[0]);
System.out.println("The value of y[1] at x = " + xn + " is
" + yn[1]);
System.out.println("The value of y[2] at x = " + xn + " is
" + yn[2]);
System.out.println("Number of iterations = " +
rk.getNumberOfIterations());

// Set values needed by fixed step size method
    rk.setToleranceScalingFactor(1e-5);
    rk.setToleranceAdditionFactor(1e-3);

// Call Fehlberg method
    yn = rk.fehlberg(dn);
// Output the results
System.out.println("\nFehlberg-Runge-Kutta procedure");
System.out.println("The value of y[0] at x = " + xn + " is
" + yn[0]);
System.out.println("The value of y[1] at x = " + xn + " is
" + yn[1]);
System.out.println("The value of y[2] at x = " + xn + " is
" + yn[2]);

```

```

System.out.println("Number of iterations = " +
rk.getNumberOfIterations());

// Call Cash Karp method
        yn = rk.cashKarp(dn);
// Output the results
System.out.println("\nCash-Karp-Runge-Kutta procedure");
System.out.println("The value of y[0] at x = " + xn + " is
" + yn[0]);
System.out.println("The value of y[1] at x = " + xn + " is
" + yn[1]);
System.out.println("The value of y[2] at x = " + xn + " is
" + yn[2]);
System.out.println("Number of iterations = " +
rk.getNumberOfIterations());
    }
    catch(SQLException se){
        se.printStackTrace();
    }
    catch(ClassNotFoundException cnfe){
        cnfe.printStackTrace();
    }
}

}

// Class to evaluate the three derivatives for given values
of x and the three y[i].
class Derivn1 implements DerivnFunction{
    private double k1 = 0.0D, k2 = 0.0D;
    public double[ ] derivn(double x, double[ ] y){
        double[ ] dydx = new double [y.length];
        dydx[0] = -k1*y[0];
        dydx[1] =  k1*y[0] - k2*y[1];
        dydx[2] =  k2*y[1];
        return dydx;
    }
    public void setKvalues(double k1, double k2){
        this.k1 = k1;
        this.k2 = k2;
    }
}
}

```

## APPENDIX B

### REFERENCES

1. <http://www.thoughtsoncloud.com/2014/03/a-brief-history-of-cloud-computing>
2. Graziano, Charles David, "A performance analysis of Xen and KVM hypervisors for hosting the Xen Worlds Project" (2011). *Graduate Theses and Dissertations*. Paper 12215.
3. Autonomic Cloud Computing: Open Challenges and Architectural Elements Rajkumar Buyya, Rodrigo N. Calheiros, and Xiaorong Li
4. A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing Georgia Sakellari, George Loukas
5. An Analytical Model for Multitier Internet Services and Its Applications Bhuvan Urgaonkar, Giovanni Pacifici, Prashant Shenoy, Mike Spreitzer, and Assem Tantawy
6. Httpperf - A Tool for Measuring Web Server Performance. David Mosberger Tai Jin, HP Research Labs, Hewlett-Packard Co. Palo Alto CA 94304 [ch.4]
7. Automated Control for Elastic n-Tier Workloads based on Empirical Modeling by Simon Malkowski, Markus Hedwig, Jack Li, Calton Pu, and Dirk Neumann.
8. Sharma, Upendra, "Elastic Resource Management in Cloud Computing Platforms" (2013). Open Access Dissertations. Paper 763.
9. VM consolidation: A real case based on OpenStack Cloud. Antonio Corradi, Mario Fanelli, Luca Foschini. Dipartimento di Elettronica, Informatica e Sistemistica (DEIS), University of Bologna, Italy
10. Experimental Evaluation of OpenStack Compute Scheduler by Oleg Litvinski and Abdelouahed Gherbi. Department of Software and IT Engineering, École de technologie supérieure (ÉTS), Montreal, Canada [ch.2]
11. OpenStack Installation Guide for Ubuntu 12.04 (LTS), havana (2014-09-10). Copyright © 2012, 2013 OpenStack Foundation All rights reserved. [ch.2]
12. <http://www.thoughtsoncloud.com/> [ch.2]
13. <http://opensource.com/resources/what-is-openstack> [ch.2]

14. <http://networkstatic.net/openstack-essex-scripted-installation-on-ubuntu-12-04-part-1/> [ch.2]
15. <http://networkstatic.net/openstack-essex-scripted-installation-on-ubuntu-12-04-part-2/> [ch.2]
16. <https://virt-manager.org/> [ch.3]
17. <https://technet.microsoft.com/en-us/library/bb963715.aspx> [ch.3]
18. <https://help.ubuntu.com/community/Xen> [ch.3]
19. [https://help.ubuntu.com/community/Setting up Xen and XAPI, XenAPI on Ubuntu Server 12.04 LTS and Managing it With Citrix XenCenter or OpenXenManager](https://help.ubuntu.com/community/Setting%20up%20Xen%20and%20XAPI,%20XenAPI%20on%20Ubuntu%20Server%2012.04%20LTS%20and%20Managing%20it%20With%20Citrix%20XenCenter%20or%20OpenXenManager) [ch.3]
20. <http://www.hpl.hp.com/research/linux/httpperf/httpperf-man-0.9.pdf> [ch.4]
21. <http://www.hpl.hp.com/research/linux/httpperf/> [ch.4]
22. [https://www.apachefriends.org/faq\\_linux.html](https://www.apachefriends.org/faq_linux.html) [ch.4]
23. Java XAMPP Connect to MySQL database part1 + Stored Procedures  
<https://www.youtube.com/watch?v=s7uP16F0Y6E> [ch.5]
24. <http://www.ee.ucl.ac.uk/~mflanaga/java/RungeKutta.html> [ch.5]
25. [http://www.pearsonhighered.com/assets/hip/us/hip\\_us\\_pearsonhighered/samplechapter/0131018159.pdf](http://www.pearsonhighered.com/assets/hip/us/hip_us_pearsonhighered/samplechapter/0131018159.pdf) [ch.5]
26. Java Connect to Xamp MySQL database  
[https://www.youtube.com/watch?v=tW86s5u\\_Cu8](https://www.youtube.com/watch?v=tW86s5u_Cu8) [ch.5]
27. <http://www.elithecomputerguy.com/>