11-16-2012

Approaches to Provisioning Network Topology of Virtual Machines in Cloud Systems

Mani Shafaatdoost
Florida International University, mshaf012@fiu.edu

DOI: 10.25148/etd.FI12120510

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

APPROACHES TO PROVISIONING NETWORK TOPOLOGY OF VIRTUAL MACHINES IN CLOUD SYSTEMS

A thesis submitted in partial fulfillment of

the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Mani Shafaatdoost

2012

To: Dean Amir Mirmiran
College of Engineering and Computing

This thesis, written by Mani Shafaatdoost, and entitled Approaches to Provisioning Network Topology of Virtual Machines in Cloud Systems, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this thesis and recommend that it be approved.

Ming Zhao

Peter J.Clarke

S.Maosud Sadjadi, Major Professor

Date of Defense: November 16, 2012

The thesis of Mani Shafaatdoost is approved.

Dean Amir Mirmiran
College of Engineering and Computing

Dean Lakshmi N. Reddi
University Graduate School

Florida International University, 2012

DEDICATION

This work is lovingly dedicated to my parents

My Mother, Firoozeh Farnia

My Father, Seyfollah Shafaatdoost

ACKNOWLEDGMENTS

This research would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study.

First and foremost, my utmost gratitude goes to Professor. S. Masoud Sadjadi, whose sincerity and encouragement I will never forget. Dr. Sadjadi has been my inspiration as I hurdle all the obstacles in the completion of this research work.

My sincere thanks also go to Professor Peter J Clarke and Professor Ming Zhao for their guidance in the evaluation part of my thesis. They have shared valuable insights in the relevance of the study.

I consider it an honor to work with Dr David Villages, former PhD student of Florida International University for bringing me a broad view of technical issues regarding my project. Despite of all of his own works for his thesis, he helped me so much during research and development of my thesis.

I wish to express my gratitude to the staff of the NSF PIRE program (Partnerships for International Research and Education) especially Dr. Scott Graham and Mr. Luis Rivera for providing me with a scholarship for going to North Carolina State University and Federal Flemense University.

I owe my gratefulness to Professor Mladen A. Vouk and Professor David R. Wright from North Carolina State University for offering me the summer internship opportunities in their groups and leading me to work on a set diverse exciting project.

I would like to gratefully acknowledge Florida International University, North Carolina State University and IBM for providing me with great equipment for doing my research and experiments.

I would like to thank Mr. Georgy Mathew Kallumkal, master student and Marhn Fullmer, the manager of network lab in North Carolina University for the assistance on how to use the hardware and software needed for my project.

ABSTRACT OF THE THESIS

APPROACHES TO PROVISIONING NETWORK TOPOLOGY OF VIRTUAL MACHINES IN CLOUD SYSTEMS

by

Mani Shafaatdoost

Florida International University, 2012

Miami, Florida

Professor S. Masoud Sadjadi, Major Professor

The current infrastructure as a service (IaaS) cloud systems, allow users to load their own virtual machines. However, most of these systems do not provide users with an automatic mechanism to load a network topology of virtual machines. In order to specify and implement the network topology, we use software switches and routers as network elements. Before running a group of virtual machines, the user needs to set up the system once to specify a network topology of virtual machines. Then, given the user's request for running a specific topology, our system loads the appropriate virtual machines (VMs) and also runs separated VMs as software switches and routers. Furthermore, we have developed a manager that handles physical hardware failure situations. This system has been designed in order to allow users to use the system without knowing all the internal technical details.

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF FIGURES

LIST OF ACRONYMS

VM	Virtual Machine
VMM	Virtual Machine Manager
VL	Virtual Lab
VCL	Virtual Computing Lab
NIC	Network Interface Card
VNIC	Virtual Network Interface Card
DB	Database
UI	User Interface
MN	Management Nodes
VPN	Virtual Private Network
VLAN	Virtual Lan
VTMCI	Virtual Topology Manager in Cloud Infrastructure
HPC	High Performance Computing
RDP	Remote Desktop Procedure
FIU	Florida International University
NCSU	North Carolina State University

1. Introduction

Several decades ago, IBM introduced the concept of virtual machines (VMs) for sharing a single set of hardware among multiple users [1] [2]. In recent years, many companies such as VMware [3] and Amazon [4] have developed software that allows multiple users to share one physical machine. Each user can use this software to load his/her own VM. Vendors and software have guaranteed isolation among VMs. However, some users require a group of connected VMs. In this thesis we introduce different methods of loading VMs and setting up connections within a group of VMs according to a predefined network topologies. In order to provide the network topology, we use software switches and routers as network elements. Each switch or router is installed on a VM and provides networking services to other VMs.

There are several solutions that would allow users to create a network of VMs with each having their own drawbacks. If we use VMware, we can run multiple VMs and create virtual networks among these VMs; however we are required to run all VMs using a specific virtualization technology (VMware). Furthermore, if we want to have a network of VMs loaded on different physical machines; we would need to acquire additional software from the vendor. Another solution is to allow users to run their own VMs and set up the network manually among these VMs. However this method requires expertise, it is time consuming and it is not modular.

In order to provide a robust solution, we have to consider few points. First, users must be able to use our system without knowing much of the technical details. Secondly, network delay must be tolerable by the application that is running on this platform. It should also be easy to migrate from one platform and technology to another platform or technologies. Lastly, this system must work with current virtualization technologies.

To design a network topology, we developed three different solutions by using software network elements on different platforms. We explore each solution and then discus pros and cons. The user will choose the most appropriate solution depending on the requirements of the application.

In our solution we install software switches and routers on VMs and use them as network elements. Before running a group of VMs, a user needs to set up the system to make a topology of VMs and specify the connections among them. Then, given the user's request for running a particular topology, we load appropriate VMs for his/her use and also run separated VMs as software switches and routers. Thus, we can run each VM by using different virtualization technologies and we do not need to bind ourselves to the specific technology or platform e.g. Xen, Virtual Box. Also, we implemented a new cloud manager that accepts different topologies as input and provides a reliable infrastructure for running a topology of VMs.

This thesis starts with a review of virtualization technologies and then we will describe currently existing software solutions. We continue by proposing three possible solutions to create a network of VMs and then compare experimental results of each solution. At the end we conclude by showing the effectiveness of our proposed solutions.

2. Background

To run an application on a computer system, we need to have software run on top of an operating system, and have that operating system run on top of the appropriate hardware. Applications and the operating systems need to run on the same hardware, and they need to share the resources among themselves. Time-sharing methods facilitate sharing hardware among multiple applications.

In order to run an application on hardware, it is mandatory to translate its code into machine code, which is close to assembly code. Following this, we can simply run each command in the processor. Where sharing multiple applications on a single processor, we simply need to define separated address spaces for saving their information and then use time-sharing to share CPU and other resources among these applications. It is possible to use simulators for simulating one application in a specific hardware system. Sometimes people use simulation for reducing software development expens.

We can specify one machine as an input for other machine. We need to describe one machine and its input and then pass them to another machine, the second machine is going to simulate running of the first machine within itself.

By using the Turing theorem, we can describe physical machines and run them on another host machine. By using time-sharing, we can run multiple physical machines on a single machine. Each of these guest machines is a VM and we need to have an application in the host machine to manage them, called the Virtual Machine Manager (VMM).

But virtualization technologies are not bounded only to physical machines. We can also share network resources among multiple users. Methods of sharing network resources among multiple users and multiple applications have had continued development over several years; however our focus will address network virtualization among virtual machines. Each virtual machine can have its own network interface cards and it may need to communicate with other virtual machines on the same physical machine or other physical machines and it may also need to provide the user with a connection to the outside world.

Virtualization has many benefits such as: It allows users to test their software or new operating systems with reduced costs; to share resources more effectively, to have several operating systems on a single physical machine; to improve security and availability. Additionally, isolation among different virtual machines is much easier than isolation among multiple applications in a multi-tasking operating system. Each operating system has thousands or millions of lines of code and it is really hard to debug or test these applications; but VMMs are usually smaller applications and it is much easier to make sure they are working correctly which leads to better security and protection amongst guest virtual machines. If we have a security issue or a fault in one of the virtual machines, we just need to restart that virtual machine and it doesn't affect the other virtual machines. Another advantage of VMs is their ability to facilitate easy migration. In processes, if we want to migrate one process from one operating system to another, we need to make sure that the destination system has all the required platforms and libraries, but yet live immigration seems impossible.

In virtual machines, we just need to copy a virtual machine from one host to another, while keeping track of online modifications, and then we can apply those modifications after migration to the new instance of the virtual machine. Also, during migration we won't lose our network connections among virtual machines.

On the other hand, there are several disadvantages associated to virtualization. By using virtualization, we usually need to sacrifice efficiency of our system resulting in a slower system compared to running one single operating system directly on a dedicated hardware. But assigning one operating system to each hardware system can be costly and it would be harder to maintain each separate computer. Another disadvantages of using VMs is to develop a good virtual machine monitor. VMM needs to know about the operating systems and hardware. Sometimes hardware and software companies don't want to share this information with another company. But during last few years, some companies has taken steps to incorporate code from Microsoft and made some changes in order to run them faster by using virtualization technologies [5].

By following this section, you will find some information about VMware server and VMware ESX/ESXI server. After that we will describe vSphere distributed switch which has been used by VMware for creating virtual networks among virtual machines. Finally, we will cover Open Vswitch as an open source application to manage network requirements of virtual or physical machines at the switch level.

2.1 VMware server

VMware server is an application that has been developed by VMware to manage virtual machines over a physical host [6]. There are two versions of VMware servers and each of them has its own features, but VMware server two is more complete and more stable than VMware server one.

It is easy to install and use VMware server one. In order to install, you have to download it from the VMware website for free and install it on your Linux or Windows machine. Then you will have a virtual machine monitor which is a thin layer of virtualization that allows the installation of multiple virtual machines on top of it. VMware server shares physical resources among these guest virtual machines by using time-sharing methods.

By using this server, we have multiple machines in a single physical machine, each machine has its own memory space and VMware server treats them as a bunch of files, so it is easy to copy one virtual machine from one server to another. Also, VMware server makes sure that our virtual machines are independent and in case of crashes, it won't affect other virtual machines. We can have different versions of windows and Linux machines at the same time in one physical resource.

These servers make it easier to load new servers in a few minutes and without the need to install new hardware, to run different kinds operating systems on different platforms by just using a single physical hardware, it increases utilization of physical machines by sharing resources among multiple virtual machines, and can save the state of one virtual machine as a single file in memory. Furthermore, it is easy to migrate virtual machines from VMware server to ESX/ESXI servers.

There are several new features in VMware server two in comparison to VMware server one. In this new server, we can have a broad range of operating systems such as Windows Server 2008 [26] or Ubuntu 8.04 [27]. It also supports 64-bit operating systems. There is a new web-based management interface which allows users to create and run virtual machines much easier. It allows users to have virtual machines with 8 GB of RAM, 10 network interface cards and faster USB 2.0 devices.

VMware server also has a command line interface which allows the user to write commands to create and run virtual machines. By using these commands we can simply develop scripts to create virtual machines, load them, stop them and delete them from the memory.

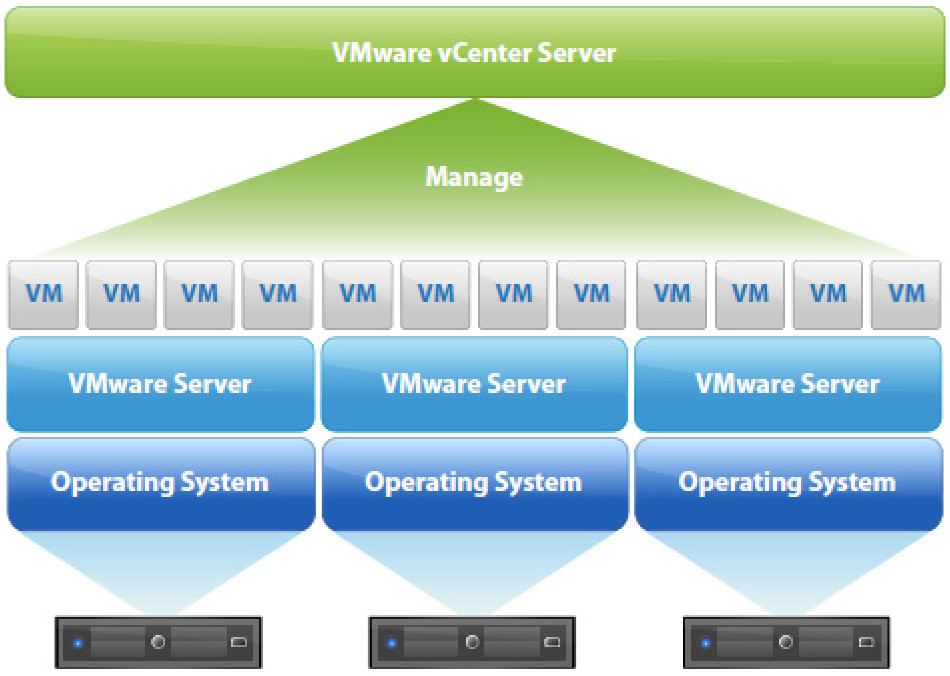


Figure 1 VMware Server partitions a physical server into multiple virtual machines [7]

As you can see in figure 1, there is one VMware vCenter Server to manage multiple VMware servers on different physical nodes [8].

2.2 VMware ESX/ESXI

VMware ESX/ESXI provides a reliable high-performance platform to run virtual machines. VMware ESX/ESXI installs on the system as an operating system. It is a modified version of Linux. Some of the mandatory commands of traditional Linux have been removed from this kernel and VMware added some specific commands for managing virtual machines and networking.

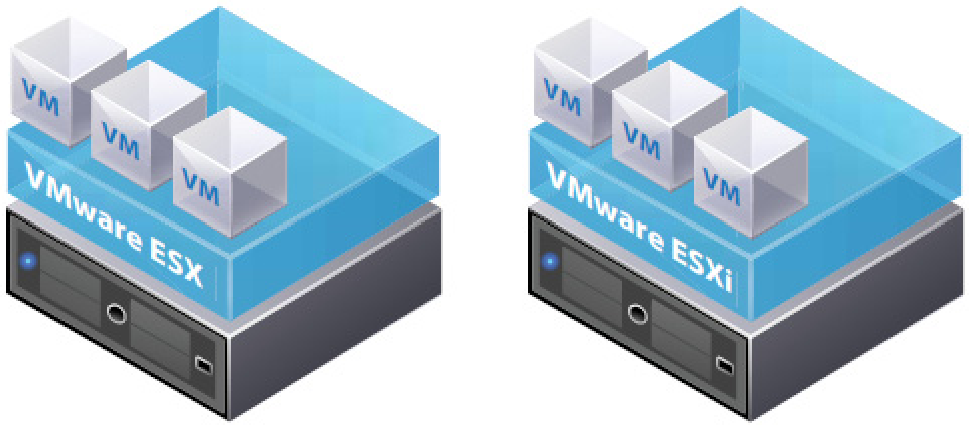


Figure 2 VMware ESX and VMware ESXi virtualized server, storage and networking [9]

As you can see in figure 2, VMware ESX/ESXI with a thin architecture bring reliable platforms to run unmodified operating systems on them. They guarantee reliability, high performance, and isolation among virtual machines.

VMware ESX/ESXI supports 64-bit architectures and up to 1TB of RAM. By supporting stronger servers which can have 1TB of RAM and 64 physical CPU, it is possible to support bigger virtual machines with up to 255 GB RAM, 8-way symmetric multiprocessing. It also allows some virtual machines to have direct access to hardware devices.

By using virtual machine disks, it is possible for virtual machines to have access to their own data in a specific part of storage device and ESX/ESXI can use that space for migration or managing hardware storage. Virtual machine file system eliminates the single point of failure issue. ESX/ESXI hosts allow users to create a network topology among virtual machines which are located inside one physical machine, and allow users to set up specific bandwidth constraints for each virtual machine.

ESX/ESXI has a new set of commands in the command line interface, which brings more flexibility and more extendibility for managing VMs. For security reasons, VMware ESX/ESXI has disabled some of traditional Linux commands. At the same time there is a graphical interface, VMware Vsphere [10]. This graphical interface allows users to connect to ESX/ESXI from remote machines, manage internal resources, manage networks, make new virtual machines, install new operating system on virtual machines, load and stop virtual machines, and more. This interface is a strong interface that can do almost everything related to managing virtual machines and physical resources.

By using these servers, we can simply migrate all virtual machines that have been developed by other virtualization technologies to the current format. In order to do so, we need to install VMware vCenter Converter which is a free software from VMware company [11]. This software allows users to convert various virtual machines to another format of virtual machines.

2.3 VSphere Distributed Switch

Vsphere Distributed Switch (VDS) has been designed to make a simplified interface for managing network requirements of virtual machines over multiple VMware ESX/ESXI hosts[12].

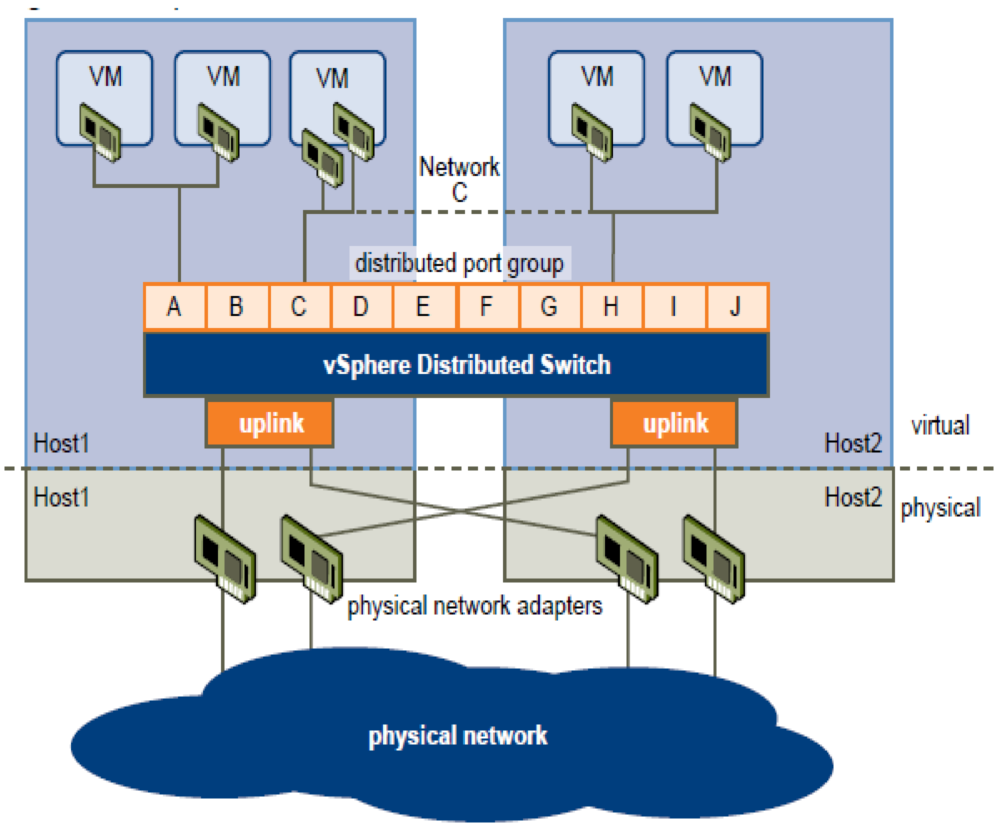


Figure 3 VMware vSphere Distributed Switch [13].

As shown in figure 3, there are multiple virtual machines in different VMware ESX/ESXI hosts; however by using VDS, we can assume that all of the VMwares are located in the same network segment. VDS allows us to manage port groups, port configurations, control link access between physical hardware and each virtual machine, and check network health.

In each ESX/ESXI server, we can manage virtual network interface card (VNIC) of each virtual machine. On each physical machine there are some physical network adaptors which are connected to the outside, and there are some VNICs for each virtual machine. By using an ESX/ESXI graphical user interface or command line interface, we can make different port groups and add each VNIC or physical NIC to each of the port groups. Furthermore, we can make a VLAN for communication between virtual machines. physical or virtual machines which are located at the same VLAN have a illusion of being at the same network physical network. VLANs can be configured in three different levels: host level, ESX/ESXI level, and hardware level .In order to make a VLAN in host level we need to have specific VNIC for each virtual machine and there should be one specific driver for each operating system, also there is ought to be different setups in each operating system. Another way of making VLAN is using port group feature of ESX/ESXI, by using graphical user interface, it is easy to add one port group to specific VLAN. Another method of making VLAN is using hardware switches. In this approach; we can setup one port to be in specific VLAN. Finally, we need to setup switches to trunk mode which allows traffic passes over all hosts.

2.4 Open Vswitch

Open Vswitch is a software switch which brings a lot of common features of traditional physical switches [14]. By using GRE tunneling, NetFlow, SFlow, SPAN, and RSPAN[14]; it provides visibility among virtual machines. It supports LACP (IEEE 802.1AX-2008), Standard 802.1Q VLAN model with trunking, STP (IEEE 802.1D-1998), HFSC qdisc, IPv6 , OpenFlow protocol and tunneling protocols. [14]

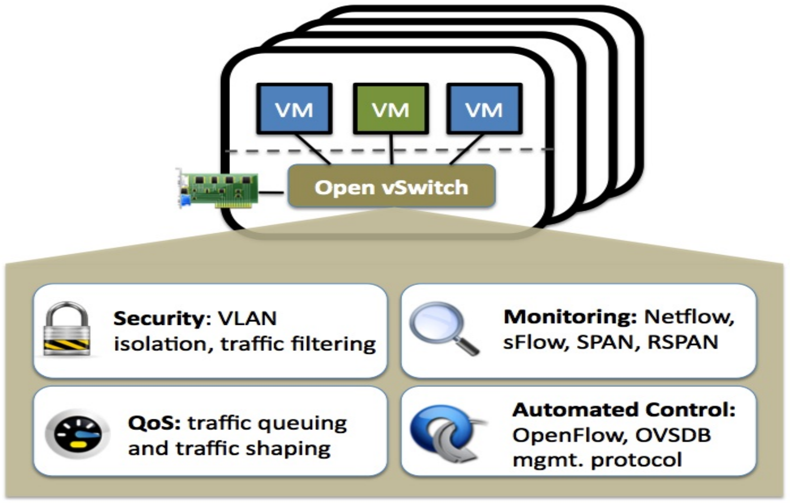

VM
VM
VM
Open vSwitch
Security: VLAN isolation, traffic filtering
Monitoring: Netflow, sFlow, SPAN, RSPAN
QoS: traffic queuing and traffic shaping
Automated Control: OpenFlow, OVSDB mgmt. protocol

Figure 4 Open Vswitch[14]

We can install Open Vswitch on physical or virtual machines. It can be installed as a separate operating system or as a kernel module. Usually when we want to test our application, we need to install Open Vswitch in the user-level. If we install this software on a physical machine, then we need to provide it with a lot of network interface cards in other to allow communication between this machine and other machines. Moreover, we can install this application on a virtual machine and then we just need to have multiple virtual network interfaces.

In some open source virtual machine managers, we have an integrated image of this software and we just need to simply load this switch, then it discovers all other virtual network interface from other virtual machines automatically [15,16].

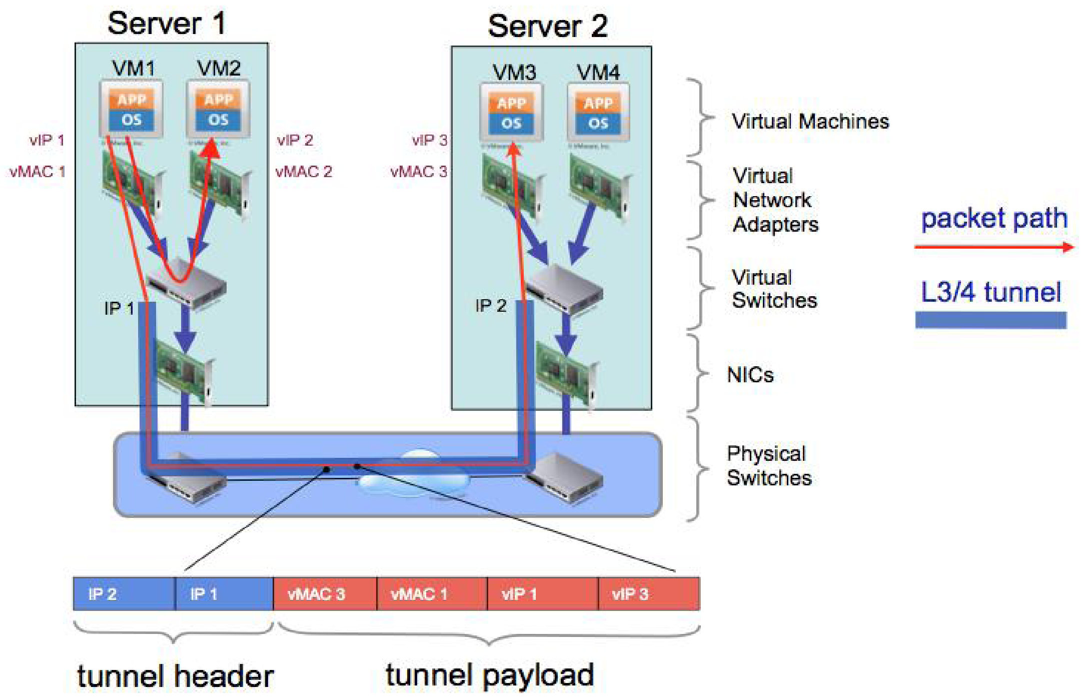


Figure 5 Communication between two virtual machine by using Open Vswitch [17]

\

As shown in figure 5, we can have one Open Vswitch on each physical host, and then we need to set up a network among Open Vswitch and virtual machines. If we use open source cloud manager software, then we do not need to worry about internal connection between virtual machines and virtual switches. On the other hand, for VMware, we need to use some tricks that we will mention later. For communication among virtual switches in different physical host, we simply need to use a tunneling algorithm. Open Vswitch supports different methods of tunneling, but usually we use GRE tunneling among multiple host. By establishing tunnel among two physical hosts we join two segments of networks to one segment. Virtual machines in both sides have no information about underlying protocol and they work as if they are in the same network.

2.5 xCAT(Extreme Cloud Administration Toolkit)

IBM released xCAT as an open source project to control datacenters [18]. This software allows users to quickly load, install, manage and provision operating systems on physical or virtual machines. xCAT is not a virtualization technology, in fact it is a software which can load operating systems on different physical or virtual machines.

xCAT allows users to easily change one physical machine from one virtualization technology to the other one. For example if we have a VMware ESX/ESXI hypervisor on our physical machines and we want to change it to a simple Windows system; it is easy to use xCAT. xCAT allows us to change these operating systems automatically by using predefined scripts.

This specific feature of xCAT makes it unique in this technology. Sometimes we have limited numbers of physical machines and we have a broad range of users with different requirements. Some users need to have direct access to a web server, some others may need to have a VMware ESX/ESXI to load multiple virtual machines on it, and other users may need to have a VMware Server to manage their virtual machines by using traditional Linux commands. For this reason, it is a good idea to have software which can manage different kinds of requirements and load different kinds of operating system or cloud manager according to the requirements.

xCAT is highly scalable and it is possible to manage up to 1000 machines by using xCAT management nodes. Moreover, users can connect different xCAT management nodes to each other and use a hierarchical model of management.

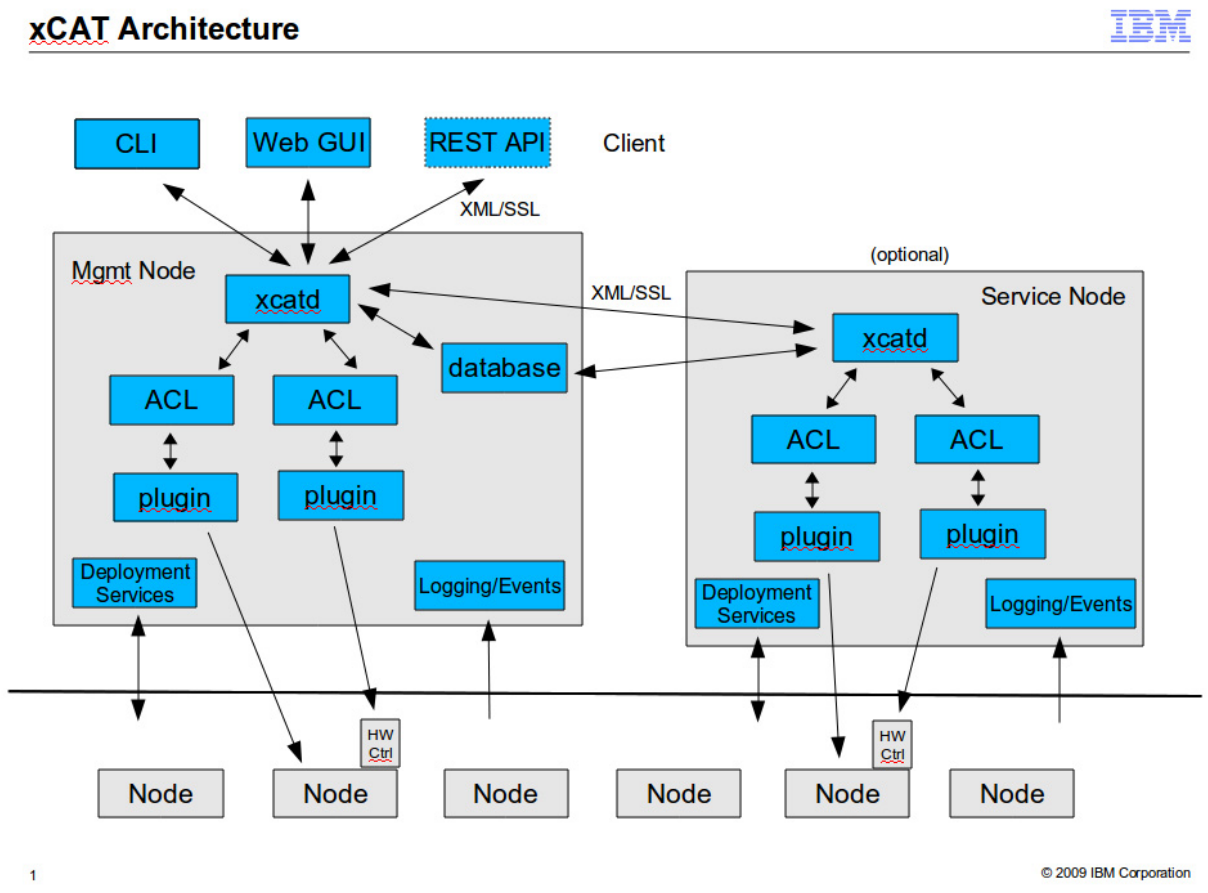


Figure 6 xCAT Architecture [19]

As shown in figure 6, the xCAT architecture consists of several parts. The client part is the collection of different methods which allow users to connect to the management node and manage xCAT. It is possible to work with a web graphical interface, which is easier to use, or just a command line interface, which is usual in Linux based software. This feature allow user to write shell scripts for different commands.

If the users want to write a program on top of xCAT to manage different virtual or physical machines, then they can use web services. In each system, there are different nodes which can be physical or virtual machines and by using xCAT, it is possible to automatically install all kind of operating system on these machines.

3. Related Work

There are different projects which have been developed to manage the interaction between users and virtual machines. These applications called cloud managers and they have control over physical machines. By using predefined methods, they can load VMs on top of these physical machines.

In this section we introduce two of these applications and compare pros and cons of each of them. Virtual Lab (VL) allows admin to design a network of VMs. Then, upon user's request, VL will load this topology of VMs. VL has been written in order to work with VMware server one. Virtual Lab has two main components, user interface which has been implemented by Moodle and provisioning module which has been developed by using python scripts and shell scripts.

Virtual Computing Lab (VCL) allows admin to manage multiple VMs and edit these VMs and save them as new VMs. Then, upon user's request, VCL loads the specific VM for the user. VCL allows switching of resources to high performance computing (HPC) during idle time.

VCL consists of three major parts, Interface, Database and Management Node. Interface is a simple web-based interface which allows users to set up and reserve virtual machines from this point of view; the database consists of several of tables of VCL's information, management node of VCL is responsible to read information from database and provision virtual machines on physical servers.

3.1 Virtual Lab (VL)

The IT-automation course is presented as one of the undergraduate courses of computer science department at FIU. In this course each student needs to have access to multiple computers via a predefined network to do the practical experiments on them. But it is not possible to assign 5 physical machines for each student. Virtual Lab (VL)[20] has been developed to solve this problem. VL brings ability for professors to create a template that consists of network of virtual machines. After that, each student can connect to the system and request for reserving one template, then the system assigns one copy of template to each student, (the network topology for each student is like figure 7).

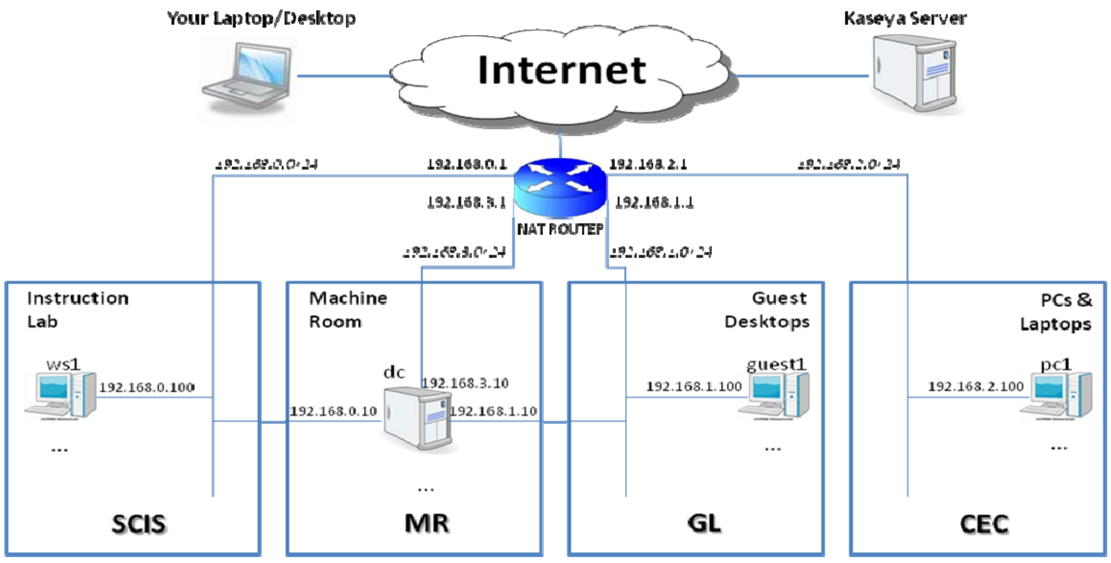
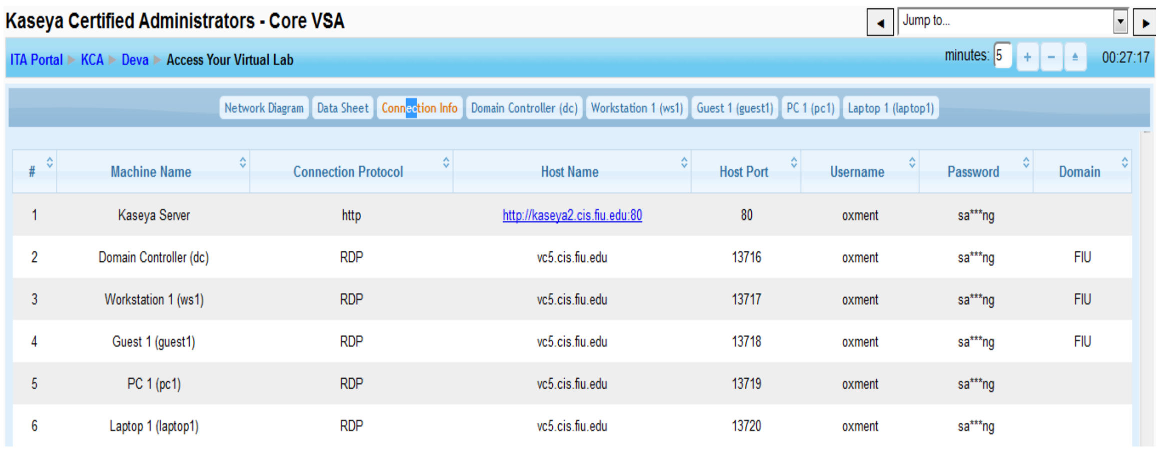


Figure 7 Virtual Lab Network Diagram [20]

Virtual Lab has two main components, user interface and provision module. We can implement each module independently and attach them via predefined protocols. For this reason, we can have different kind of user interfaces. Each user interface has access to a provisioning module and can send different messages to load different virtual machines and networks among them. Current implementation of VL is using Moodle as an interface.

By using Moodle, there are two general methods for using resources, on-demand or by reservation. When we need to have access to our virtual machines, we just have to login to the Moodle and click on the VL link. But to be sure of the availability of resources, we can use a reservation system to reserve resources before connecting to the system. Figure 8 shows the situation of the system after provision. As you can see, students can use web remote desktop procedure (RDP) and connect to destination VMs.



#	Machine Name	Connection Protocol	Host Name	Host Port	Username	Password	Domain
1	Kaseya Server	http	http://kaseya2.cis.fiu.edu:80	80	oxment	sa***ng	
2	Domain Controller (dc)	RDP	vc5.cis.fiu.edu	13716	oxment	sa***ng	FIU
3	Workstation 1 (ws1)	RDP	vc5.cis.fiu.edu	13717	oxment	sa***ng	FIU
4	Guest 1 (guest1)	RDP	vc5.cis.fiu.edu	13718	oxment	sa***ng	FIU
5	PC 1 (pc1)	RDP	vc5.cis.fiu.edu	13719	oxment	sa***ng	
6	Laptop 1 (laptop1)	RDP	vc5.cis.fiu.edu	13720	oxment	sa***ng	

Figure 8 Virtual Lab for Kaseya Certificated Administrators[20]

3.1.1 Provisioning module

Provisioning module has been written to manage virtual machines and network among them. It needs to have specific protocols for communication with the user interface and virtualization components. VL provisioning module has been written using Python scripts and it works with VMware server. To use this component, we need to install one Linux machine and then install VMware on top of it and finally we need to install VL on the Linux. By using Linux commands and shell scripts, this module communicates with VMware server and Linux machine, it also uses SSH for setting up networking properties of software routers. VL has been written in order to work with VMware server one and this virtualization software is pretty unstable and causes a lot of errors in the software. When managing VMware server one, we can simply type commands in Linux command line interface and run or stop virtual machines by using these commands. Because of the instability with VMware server one, it is required to add extra codes in order to support exception of the provisioning module of VMware. There are several functionalities inside VL, but for the external interface, we have two main methods, one method for running virtual machines and the other method is for stopping them from running. Table 1, shows the arguments of start command of VL. For loading a new network topology for each student, we need to have some information about each student and each request. We need to know the name of the owner, name the template or topology, port number for RDP which is going to be available for the student, number of ports, number of routers, storage ID which is associated to the students template and register action which is useful to identify the current action.

Table 1 Start Command in Virtual Lab

Start_ve.py	Command for provision one template
Mani	Owner Name
K2_1	Template Name
15061	Port Number
5	Number of Ports
1	Router ID
4	Storage Number
2	Register Action

For loading virtual machines in VMware server, we need to define the hardware of each virtual machine in a file called VMX. This file has been used to store information of hardware, network items and image address of each VMware. Since VL is supposed to load many virtual machines in each physical host, then it has to change the Mac address of each virtual network interface of each virtual machine before running, so it must assign one specific Mac address for each VNIC.

Due to instability of VMware server one, we have to change some parts of code and make VL compatible with VMware ESXI server. However, in ESXI server, we cannot use traditional Linux commands, so we put the whole VL in a separate virtual machine and send commands remotely to ESXI for running or stopping virtual machines.

3.2 Virtual Computing Lab (VCL)

3.2.1 Web

The VCL Project has been implemented by North Carolina University which allows users to reserve and use virtual machines for a period of time [21]. But VCL doesn't support network configuration among virtual machines and it is not possible to define network topology among virtual machines. Also, VCL allows switching of resources to HPC during idle time. Currently VCL is under development as an open source project of Apache.[22]

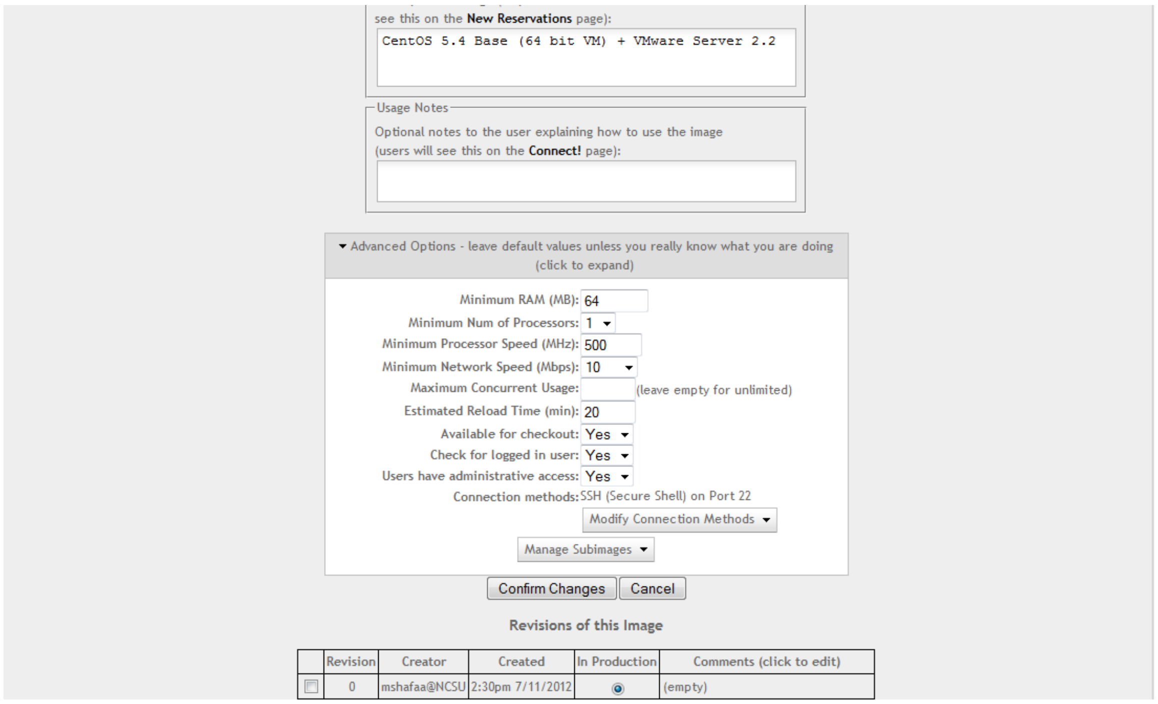
This project originally developed by North Carolina University, but today it is an open source project of Apache and IBM also works on some parts of this project [22,23]. It is possible for different institutions and organization to install a version of VCL on their local servers or simply connect to one instance of VCL and use it. Currently, most of North Carolina State universities are using VCL for their requirements [24].

VCL consists of three major parts, Interface, Database and Management Node. Interface is a simple web-based interface which allows users to set up and reserve virtual machines from this point of view. But according to our requirements we may have different views of interfaces. Here we are going to explain the simplest interface which has been designed for end users and also one other interface for admin users which allow them to manage everything related to the VCL.

The user interface is a simple interface with few pages. User can come to the system and reserve the most suitable virtual machine according to his/her own requirements. Also there is one tab which shows current reservation of one specific user.

Admin level user interface consists of a lot of information which allow admin to add new physical machines to the running version of VCL or change the computer information of each system, make new images, change current images, change networking setup of components, add new management node, edit information of current management nodes, change the schedule of resources, change user group and change resource groups.

But by using this interface we can change the information of one image in order to satisfy our requirements. As you can see in Figure 9, we can change hardware information of a specific image and add some sub images to the currently running image.



see this on the **New Reservations** page):

CentOS 5.4 Base (64 bit VM) + VMware Server 2.2

Usage Notes

Optional notes to the user explaining how to use the image (users will see this on the **Connect!** page):

▾ Advanced Options - leave default values unless you really know what you are doing (click to expand)

Minimum RAM (MB):	64
Minimum Num of Processors:	1
Minimum Processor Speed (MHz):	500
Minimum Network Speed (Mbps):	10
Maximum Concurrent Usage:	(leave empty for unlimited)
Estimated Reload Time (min):	20
Available for checkout:	Yes
Check for logged in user:	Yes
Users have administrative access:	Yes
Connection methods:	SSH (Secure Shell) on Port 22

Revisions of this Image

	Revision	Creator	Created	In Production	Comments (click to edit)
☐	0	mshafaa@NCSU	2:30pm 7/11/2012	●	(empty)

Figure 9 Change Image Profile Interface

Another important part of the interface is the scheduler. When a user comes to the system and reserve an image, all processes of scheduling and resource allocation will be handled by user interface component.

3.2.2 Database

All information in VCL is stored in a database. When VCL receives a new request for a reservation, it needs to retrieve information of current resources and then by using the scheduling module, find appropriate physical resources to run the new VM and then put this information in a table in the database.

The database consists of several of tables of VCL's information. Handling this information put a high load on database and sometimes it is necessary to separate database, user interface and management node by physical machines. There are more than 30 tables on this database.

There are different kinds of tables in the database. Some tables are just used for saving static information such as names, while other tables are necessary to save dynamic information of running versions of VCL. After setting up VCL for the first time, we need to put information of physical hosts into the database. During running VCL and on each transaction, VCL needs to restore some data from database and save back some results in it. Also, this database consists of one log table which has been used for debugging the program. Since, VCL is a multithread application, for each transaction it restores the number of last log in the log table and then increment this number by one and put the new log into the next row of log table.

3.2.3 Management node

Management node of VCL is responsible to read information from database and provision virtual machines on physical servers. Figure 10 shows the Management Node of VCL which is responsible to retrieve information from database and call appropriate functions from the cloud manager software. Cloud manager software can be a different version of VMware or XCAT. Also for making each virtual machine ready for users, management node is responsible to do some set up and run some scripts in each operating system.

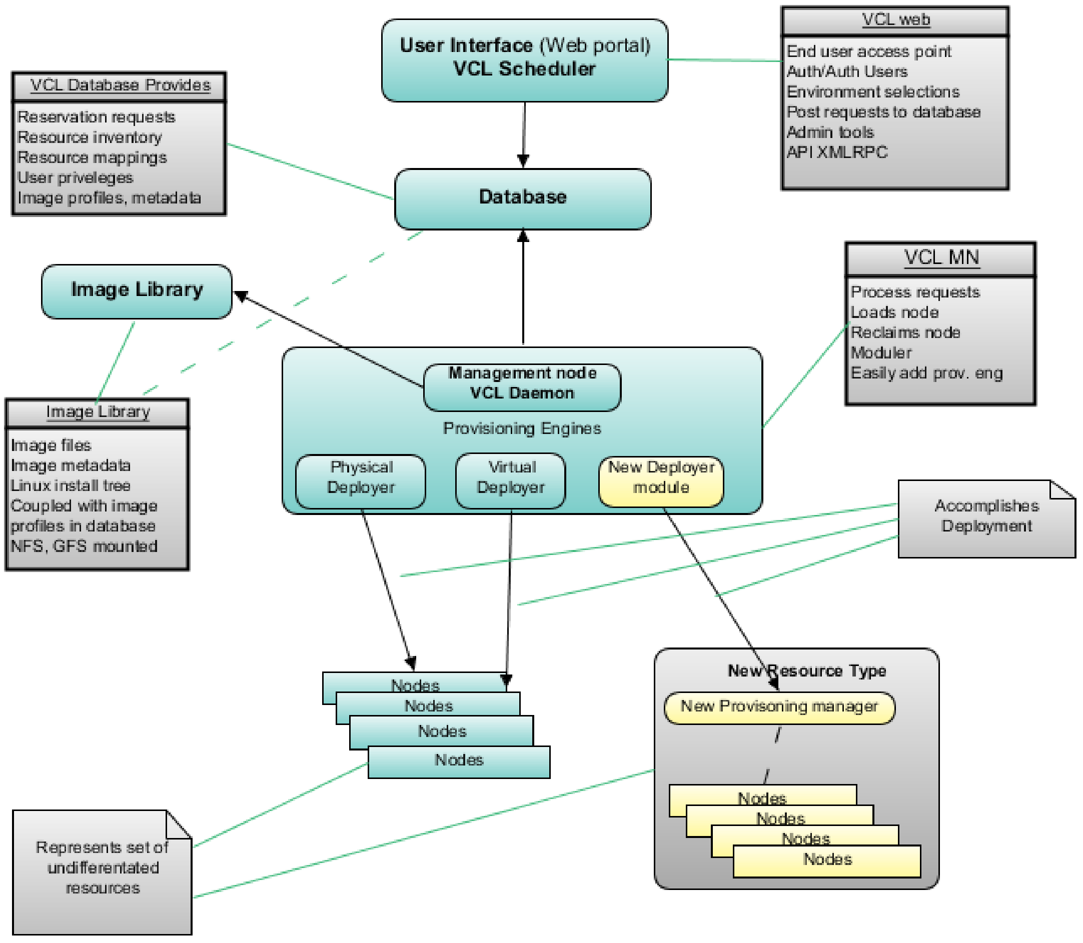


Figure 10 VCL Architecture [25]

For satisfying these requirements, the management node needs to check the database constantly for new updates. There is one main loop in VCL management node which loops every 12 seconds and it is configurable variable of VCL. This loop goes over all reservation list and check the status of each of them. It can fetch requested images from other physical servers or preload these VMs for reserved request.

Also this loop is responsible for checking images and sub-images. It makes sure that if one image has some sub-images then it can load these sub-images at the same time as it loads the main image. But there is no control over the place of loading. In fact scheduling and resource allocation has been done by the user interface part and this module only loads virtual machines.

For loading each image, management node search to find the image in its own data storage, if the image is not in there, then management node sends request to other data storages and download requested image from them. Then management node sends requests to the VMware server to load this image. After loading the image, management node loads the operating system and runs some scripts in the operating system, theses scripts makes authentication information for VCL users. Management node also runs some scripts on the hosted machine which allows VCL to send messages to the VM and receive information of the VM. Finally it creates information required for remote desktop procedure connection.

4. Automation Topology of Virtual Machines

There is an IT-automation course at (Florida International University) FIU, in which each student needs to have access to multiple computers via a predefined network to perform experiments. But it is not possible to dedicate 5 physical machines to each student. VL (Virtual Lab) has been developed to solve this problem. VL provides the professor with the ability of creating a template that consists of a network of virtual machines. After that, each student can connect to the system and request for reserving one template. And then the system assigns one copy of template to each student.

There is a VCL project that has been implemented in North Carolina State University [21] which allows users to reserve and use virtual machines. But VCL doesn't support network configuration among virtual machines and it is not possible to define network topology among virtual machines. Also, VCL allows switching resources to HPC during idle time. Currently VCL is under development as an open source project of Apache.[22]

Our goal in this project is to implement new components in VCL to allow users to make a template for a network of virtual machines. In addition, provide users with the ability to use this template and save virtual machines for future use.

Florida International University is going to use VCL as a solution for university IT services. VCL allows the university to share resources among all students and all classes and also it is possible to switch these resources to HPC at idle times. But it is not possible to use VCL for handling some courses like IT-automation. I am going to implement new components to solve this problem.

In order to satisfy these requirements we need to provide a list of possible network topologies. Then we will show that our solutions modify open sources software switches and routers and allow users to identify the following network diagrams. Possible scenarios that may occur for network requirements are shown in following pictures.

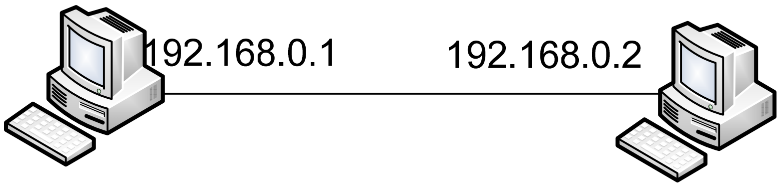


Figure 11 Direct link between two VMs

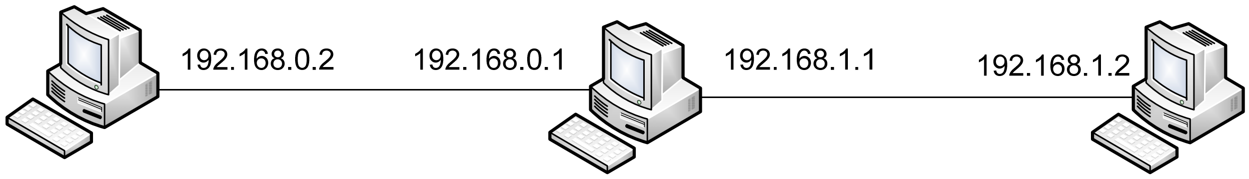


Figure 12 Direct link between two VMs and one VM may have more than one VNIC

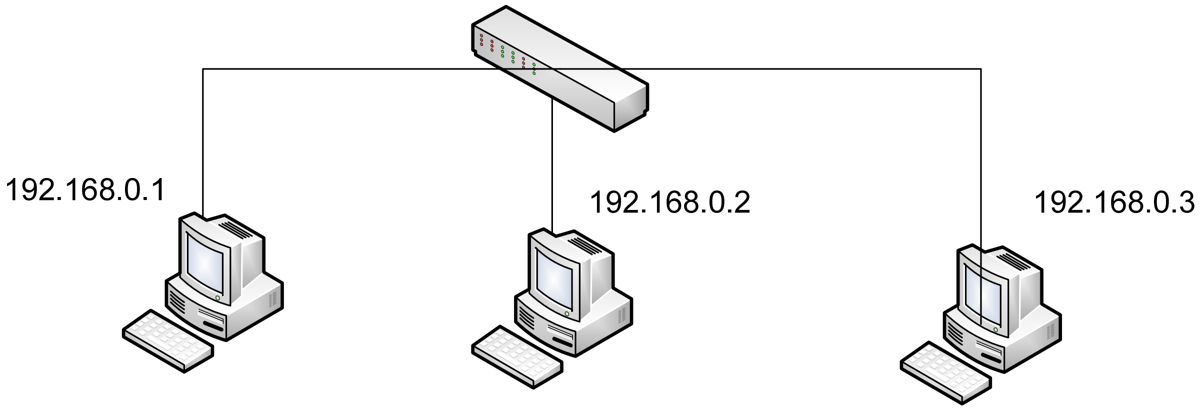


Figure 13 Three VMs with the same subnet are connecting to each other via a switch

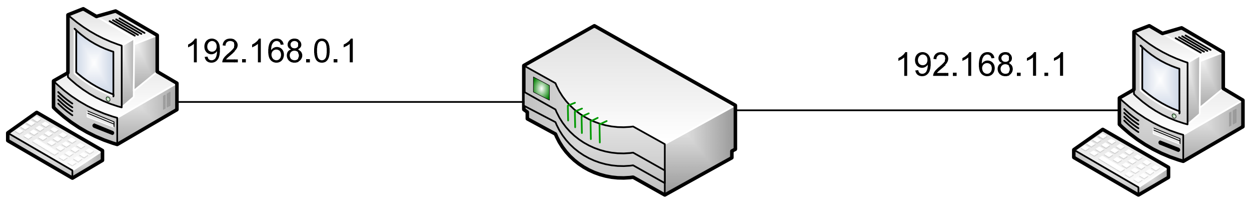


Figure 14 Two VMs at different subnets are connecting to each other via a router.

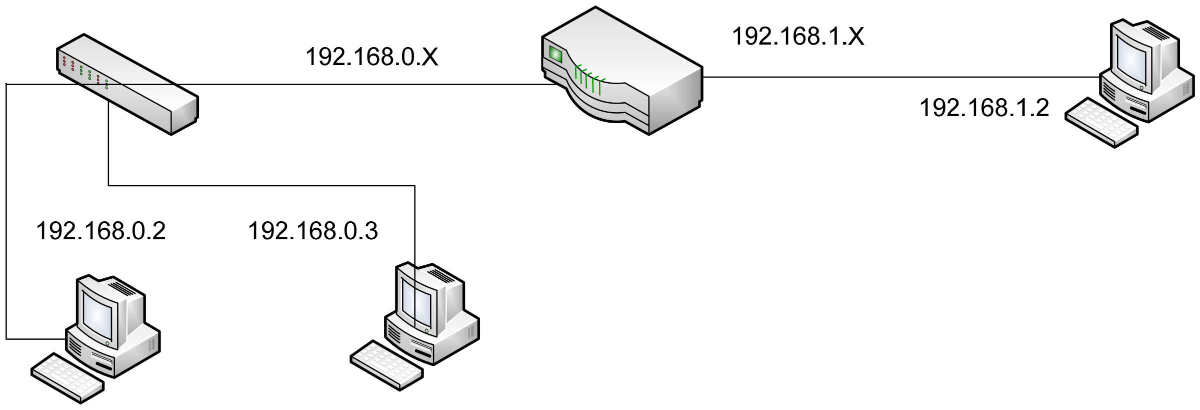


Figure 15 Three VMs from different subnets connected by switch and router

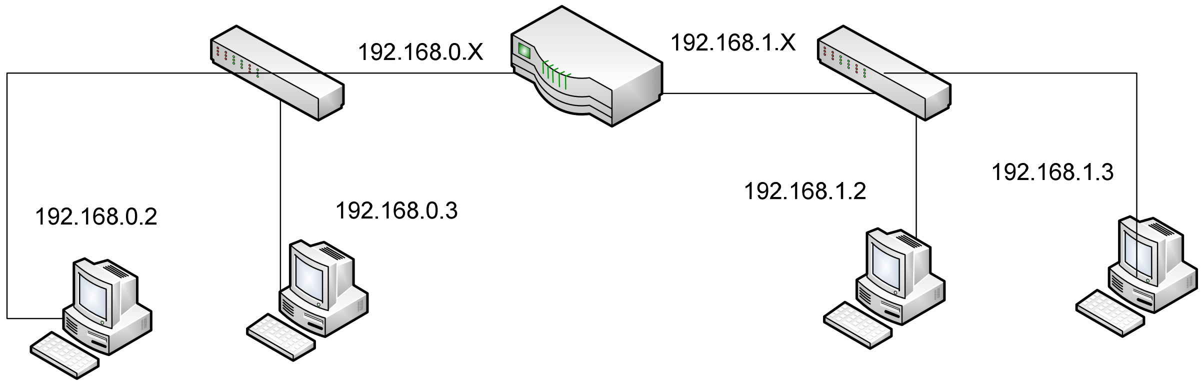


Figure 16 Four VMs from different subnets connected by switch and router

In the following sections, we describe our solution to be used for load a group of VMs by using software switches and routers. we show how to load a group of VMs by using our solutions and connect them in one of the mentioned forms.

4.1 Bare-Metal Image

By using the xCAT cloud administration tool it is possible to load a bare-metal image on the physical hardware. This feature allows users to run an image directly on a physical machine without any virtualization layer.

Cloud manager is software which provides users and developers to have access to predefined functions to manage VMs and communicate with hardware and VMs without having knowledge about the underlying hardware. For enabling networking among VMs on a bare-metal image, we can simply use VL. VL is software that uses VMware server two to provision and load virtual machines and then by using the virtual network for each topology separate them from one other. VL also uses a Linux machine as a router for handling internal requests. Unfortunately, VL doesn't know anything about hardware and it has been designed to work on specific hardware. When we run VL, it loads a predefined number of VMs on each physical machine. But as we know, a cloud manager has its own scheduling and resource allocating algorithms and it may assign VL to different physical machines. We need to add a new component to VL to determine the hardware specifications and change the maximum allowable number of VMs for this instance of VL.

In order to satisfy this condition, we have written a new script in VL which captures the information of physical machines and guest topologies. According to this information, it decides how many topologies of VMs can be loaded on this machine and changes the maximum available topologies to this number.

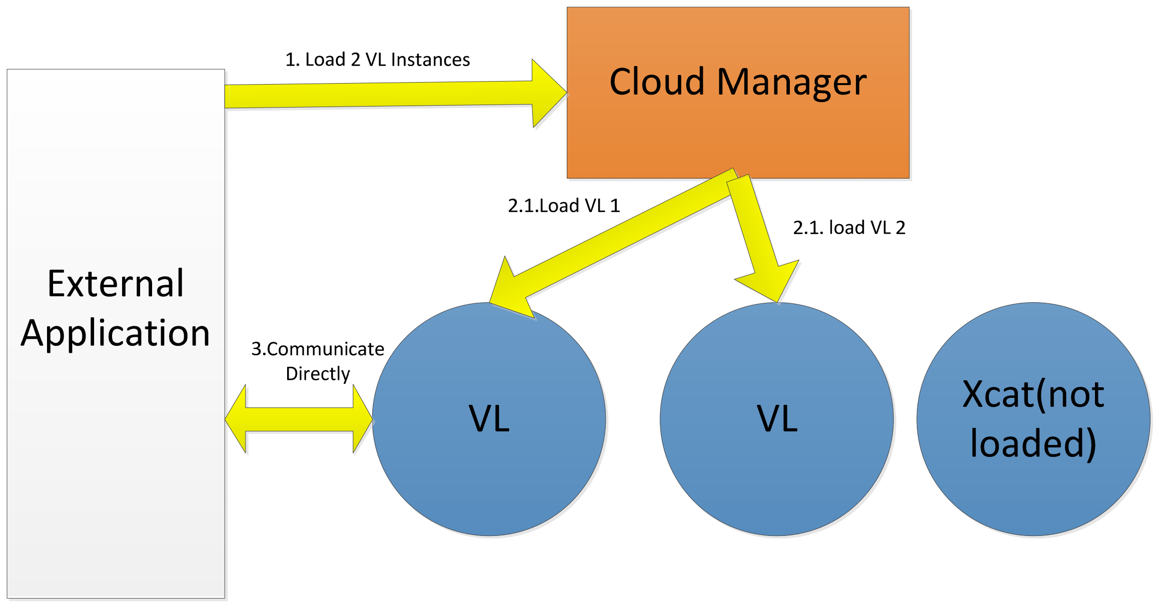


Figure 17 Load VL as Bare-Metal image

As you can see in figure 17, there are several calls that come from external application to mange internal VMs of this physical machines. In our external application, first we send a message to cloud manager to load a new VL component in the cloud system. After loading this VL, it will calculate the maximum number of topologies that we can load on this component and set this value. After that our external application can connect to the VL component and run other commands such as loading a new topology or stop running topology.

In order to manage multiple VL instances at the same time, we need to write an external proxy application to manage these items. We need to keep track of running VLs at our external application and decide to load new VLs or stop currently running VLs.

There are several advantages in loading VL as a bare-metal image. By using this method, we don’t need to think about internal interaction between VL and VMs and also this component is already written and has worked for several years without any problem.

But there are few limitations associated to this solution. For working with this method, we need to have a cloud manager which has access to several physical machines with XCAT installed on them. Also we need to know about VL and change the topology of VMs according to our requirements.

For developing this feature we developed a new component which is responsible to communicate between current VL interface and VCL interface. First we made an image from VL and put it in the VCL database. Then by using VCL's web services, we wrote a proxy class which takes request from VL and put it in an appropriate form and send it to VCL. This class has been written in PHP and consists of 200 lines of code.

4.2 Use VL as an Image in another Cloud Manager

It is possible to make a topology of virtual machines in Virtual Lab environment and save it as one image in other cloud manager systems. In this model, first we install Linux on a VM in our Cloud Manager environment. Then we need to install some packages in order to load Virtual Lab scripts in this image. Then we just need to copy our images to these VMs and make a topology among VMs of these images. Finally, we save this image as a new image in our Cloud Manager system.

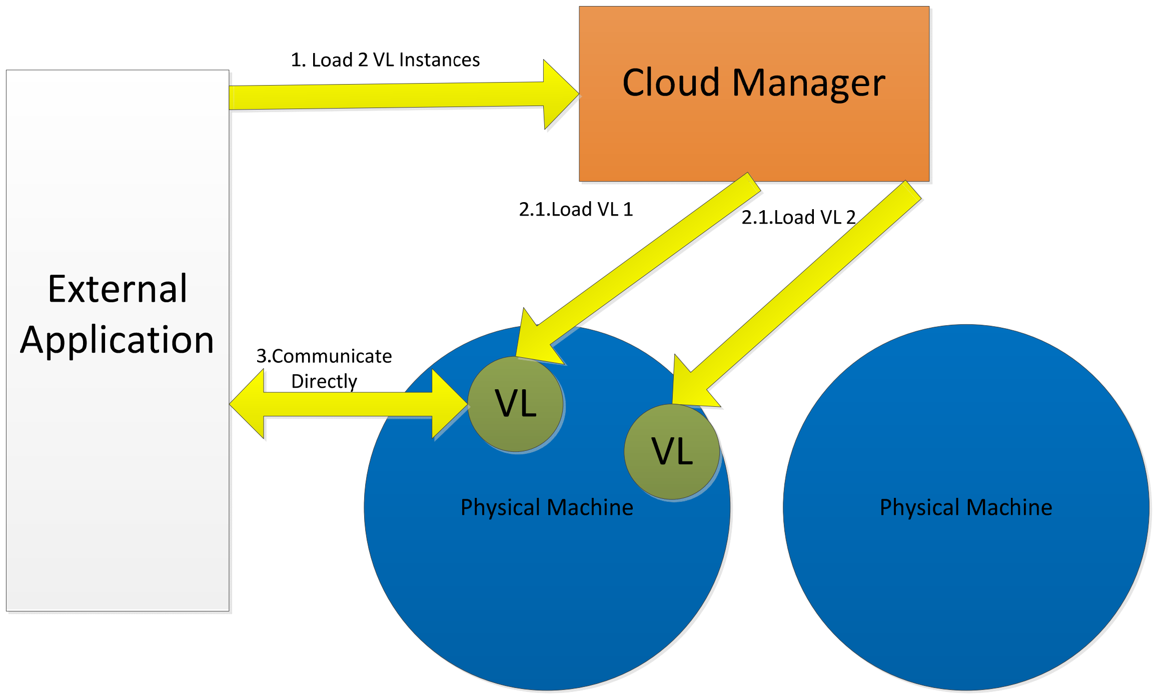


Figure 18 VL on top of Another Virtualization Technology

As you can see in figure 18, when a user requests a specific network template from the system, the Cloud Manager loads this image which has VL software on it. VL automatically loads other images internally (by using VMware Server) and then run scripts to make a network among these virtual machines. After that, the user can connect to each of these VMs in VL by using RDP.

There are a few problems associated with this solution. First of all it suffers from network delay and RDP latency but it is tolerable. Second, Cloud Manager needs to allocate one strong VM for loading VL on it (VL has more than one machine on it). Third, user needs to know how to setup a network topology of VMs on VL. But this system allows users to load VL image anywhere and it doesn't need to know about the underlying technologies or platform. It is easy to copy this image and use it in different physical machines which use different technologies and different managers to manage their physical resources.

For implementing this part, we changed VL code and removed all features which were associated to loading multiple networks. In this code we don't need to have different ports for different users, so we just load VMs and put the routing table inside the VL's VM and used this VM as a router. VL code has been written in python and for this feature; we have change about 100 lines of python and shell scripts.

4.3 Integration of Open Vswitch and VCL

Integration of Open Vswitch and VCL requires changes in VCL. Open Vswitch allows us to have a software switch on a single virtual machine and then we can connect different virtual network interfaces from other virtual machines to this software switch. Since, VMware is a closed source application; we do not have a method for determining different VNICs of other VMs. So we used another feature of ESX/ESXI which allows us to put some ports in the same port groups. When we have some ports in the same port group; they will communicate through those channels and all of the traffic in those ports will be visible to other ports on the same port group. Therefore, if we want to connect one VNIC of one VM to Open Vswitch; we just need to put these two VNICs at the same port groups and then all of the incoming traffic from that VNICs will go through the Open Vswitch. There is a specific limit for port groups which is 1024 and it is far more than our requirements for each physical machine; however, we have some limits on the number of the VNICs of each VM. Depending on the different kind of VMs, we have limited number of VNICs for each VM. This number can be varying from 8 to 16[12].

For having connection among different VMs in different hypervisors, we need to have at least one Open Vswitch in each host and this switch can use tunneling to communicate to the other switches in other hypervisors. But if we need to connect Open Vswitch to many VMs in one host; we must load more than one instance of Open Vswitch in each host.

By using Open Vswitch, we can load different types of topologies in VCL. VCL can load different numbers of VMs in different hypervisors. The only problem is the communication among VMs in different hypervisors. By having one Open Vswitch in each hypervisor, we just need to connect VNIC of that VM to Open Vswitch using port groups. At that point we would make a tunnel among those two Open Vswitches. As shown in figure 5, when a tunnel among two physical hosts is established; we join two segments of networks to one segment. Virtual machines in both sides have no information about the underlying protocol and work as they are in the same network. In order to implement this feature in VCL, we need to make some changes in different parts of the VCL code. As shown in figure 19, there are two different scenarios for adding and using a topology to the system. First a privileged user comes to the system and makes a topology to the system and we save this topology in the database. Then one normal user can come to the system and request a reservation of this topology. We need to save some runtime information of different images and then in the management node, we retrieve this information and make a network among those virtual machines.

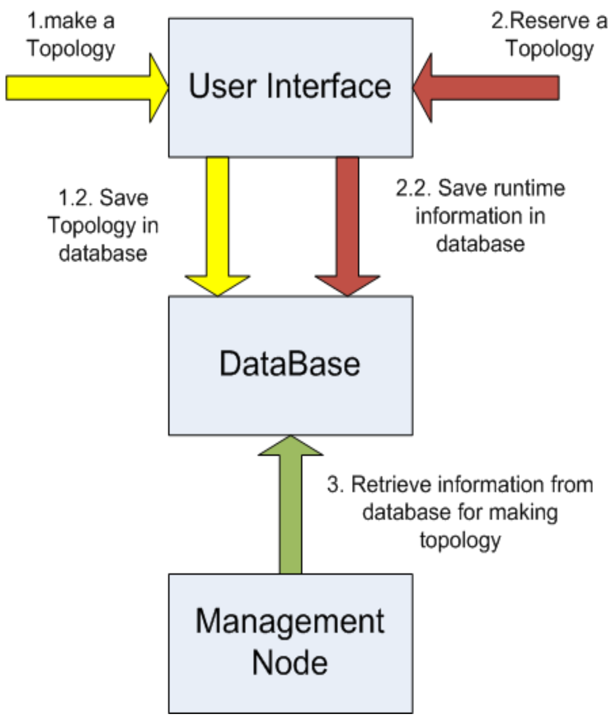

1.make a Topology
User Interface
2.Reserve a Topology
1.2. Save Topology in database
2.2. Save runtime information in database
DataBase
3. Retrieve information from database for making topology
Management Node

Figure 19 Scenarios of Making Topology and Reserving Topology

In order to add these features, there are some code changes needed in interface, database, and management node. In the interface we need to add an order to sub-images and another menu for making topologies. In the database we need to have several new tables for storing a template, information regarding reservation and topologies, Open Vswitches information and Virtual LANs (VLANs). Also we need to add one extra columanagement nodefor the image table, request table, and reservation table. Furthermore, in the management node we have to develop new scripts for loading virtual topologies, changing VMX files before loading, and adding new module for managing Open Vswitches.

In order to make a topology in UI, we need to make one image and add some sub-images to it. Consequently, VCL will load all these images together. Then using another menu, user can make a topology among the specific image and sub-image.

In database, we have a table to save all information about current topologies and Open Vswitches. Then for each request, we have to save information of the images and destination hosts. Management Node needs this information at provisioning time to make a network among these VMs. For each new request, we have to put all the VMs that are belonged to the same topology to the new VLAN, which separates them from all other instance of this topology. In order to save run-time information regarding these images, we have to add a topology ID to the master image and also to the request which consists of master image and its sub-images. Similarly for saving order of sub-images, we need to save information as a new columanagement nodein reservation table.

In management node, for loading a new image that has some sub-images and a topology ID. We also need to call a script to manage all of the network requirements of this topology. Before loading images, we have to change VMX files to add more VNIC on each VM, and put these VNIC in an appropriate port group with the other ports from Open Vswitch on the same hypervisor. For managing all Open Vswitches, we have another module which records all information and status of Open Vswitches. This module is responsible for loading one Open Vswitch in each hypervisor, make GRE tunnels among all OpenVswitches, and add new rules for making new topologies.

By using Open Vswitch, we can easily load different kind of topologies and also control the bandwidth and traffic among different virtual machines. Furthermore, it is possible to load another Open Vswitch by using other virtualization technologies in another platform and make a connection among VMs of VCL and VMs which has been loaded by other technologies.

For implementing these features in VCL code, we have changed different codes in each module. First we need to allow user to have multiple VNIC on each VM, so on the capture provisioning module (sub capture), we have edited 50 lines of code to allow users to save number of VNICs in the new row of the database in the image table. Then we have developed a simple user interface by using PHP and html code to allow user to design a network of VMs and finally we save this new template in the database. A new component in the user interface part which consists of 500 lines of PHP code allows users to store a new template and we add a new table to the database for storing topologies.

After storing a new topology, for provisioning that topology, we stored run-time information of user request. There are two new columns in request and reservation table for saving this information. Also, we modified 1000 lines of code in newReservation and submitRequest functions of Requests.php to save information into the database.

At the provisioning part, we added a new module consist of 2000 lines of PERL code to manage Open Vswitch and a table to manage the current status of Open Vswitches. When provisioning module checks for current request and starts deploying them on physical machines, it will call load function to load VMs. In this function, we call create topology method from Open Vswitch module to make a topology and store topology's information into the database.

VCL code by itself consists of 35 PHP files, more than 40 of tables in database and more than 70 PERL files. For developing new features in the VCL, we have added 2 new components and more than 2500 lines of code, and still we need to develop more codes to make sure that everything is working fine. For having more than one Open Vswitch in each physical machine, we must change the Open Vswitch component and add more information about each image of Open Vswitch.

5. Experimental Results

In this part, we have done some experiments to measure different factors which affects on performance and efficiency of different systems that we have implemented. We have done following experiments:

1. Running time of an application which has a loop for 20000000000 operations.
2. Measure latency between virtual machines and an external host.
3. Measure download time of an image of Ubuntu.
4. Measure latency among virtual machines.

For all experiments, we have used a Dell Power Edge T610 with Intel® six-core Xeon® processor 5500, VMware ESXI hypervisor, 192 GB of memory, 6 TB of internal storage and Dual Port 10GB Enhanced Intel Ethernet Server Adapter X520-DA2.

On this server, we had a virtual machine with CentOS 5.4(64 bit) and installed VMware server two on it. This virtual machine has Intel® Xeon® processor 3000, 4 GB memory, two 100MB Intel Ethernet Card.

Also, there are multiple virtual machines with 512 MB of Memory and one network interface with Ubuntu Operating system. Some of these machines have been equipped by Open VSwich as a software switch.

For the first test, we wrote an application in C to run a loop for 20000000000 and then measure the running time in virtual machine which have directly working on ESXI hypervisor and also virtual machines who have been running on top of VMware server two.

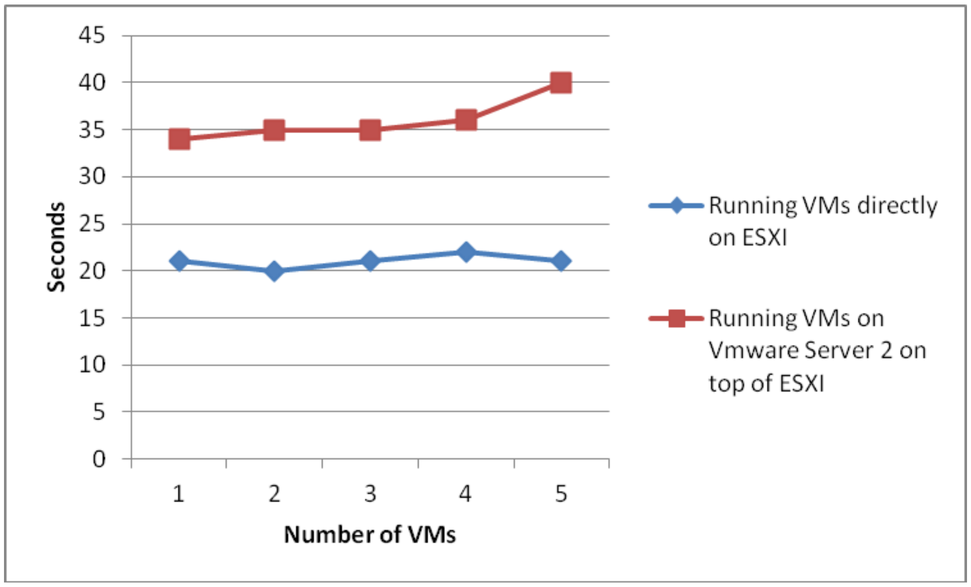


Figure 20 Running application on VMware server two and ESXI

As you can see in figure 20, the result of running an application both systems are similar and there is just small difference in the running time of this CPU intensive application by using VMware server two and ESXI. This results show that VMware server two doesn't add a significant overload in running new applications.

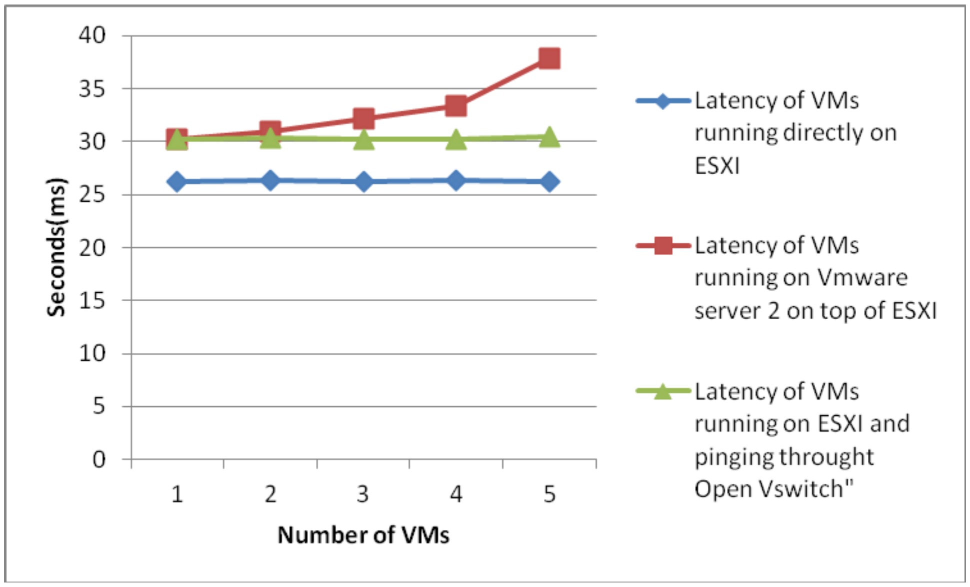


Figure 21 Latency of pinging DNS server

As shown in figure 21, there are three different curves associated to each implementation. Blue curve shows the latency of running virtual machines directly on ESXI machines and using VMware Vswitch for routing packets towards the network. Red curve shows the latency of running virtual machines on top of VMware serve two which is a virtual machine on ESXI server. Green curve shows the latency of pinging DNS server from virtual machines running on top of ESXi and pinging through Open Vswitch. It is obvious that the latency of pinging from VMs on ESXi is less than Open Vswitch and VMware Server two. But while we increase the number of VMs in VMware server two, we'll reach to saturation point which is 5 for described system. Then, by pinging from VMs on top of ESXI and using Open Vswitch delay will increase but since we the network is not overloaded, this amount is not significant.

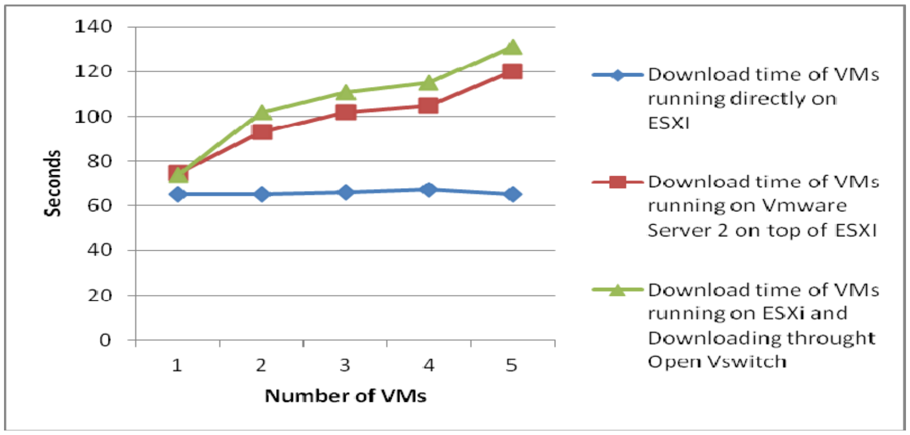


Figure 22 Download time of Ubuntu image

Figure 22 shows the download time of an Ubuntu 11.10 image from a repository which is located on east coast. Download time of these images in ESXI doesn't change significantly by increasing the number of VMs but in Open Vswitch and VMware server two, we reach to the saturation point and increasing the number of VMs after a point affects directly on download time of the image.

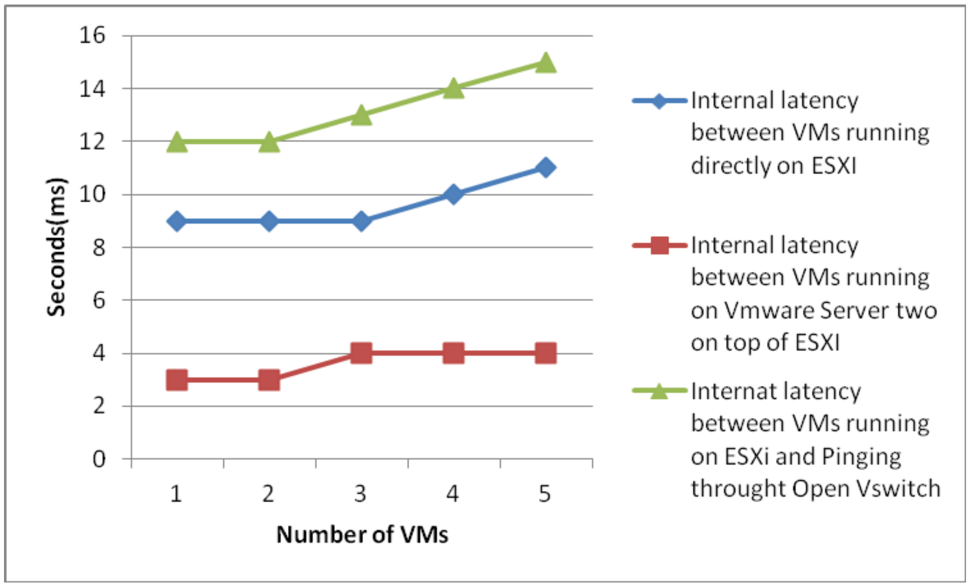


Figure 23 Latency of pinging DNS server from VMs in overloaded network

Figure 23 shows the difference of latency between internal VMs when the network is overloaded and as it shown this latency on VMware server two is less than VMware ESXI and Open Vswitch causes a greater delay on the network among two VMs.

6. Virtual Topology Manager in Cloud Infrastructure (VTMCI)

Virtual topology manager in cloud system (VTMCI) is a tool for managing infrastructure in cloud systems or over multiple cloud systems, it is also provides an environment to customize or implement new ideas for managing cloud systems. This system provides user with the ability to submit his/her request in cloud interface and change this request over the time. It also supports fault tolerance of our hardware systems. This system has components to communicate with other cloud infrastructure and send information on them.

In general, this system provide a user with a mechanism to submit a group of virtual machines with their specifications, consists of CPU and Memory requirement and also the links between these virtual machines by defining the latency and bandwidth of each link. After submitting requests, a user can change the request at anytime according to new requirements. As a cloud manager, it is possible to use this tool to manage cloud infrastructure. For each cloud system there is a management node to manage each virtual machine and submit them to the appropriate infrastructure. But as an administrator of cloud infrastructure, it is possible to use this VTMCI over other resource management in cloud infrastructure. Also, VTMCI consists of separated parts and as a cloud infrastructure admin, it is possible to configure or rewrite each of these parts according to the usage.

VTMCI consists of 3 major parts, first part is called Cloud Interface and responsible for communicate with user and probably other cloud managers.

Second part and most important part is Cloud Manager, this part itself consists of four major parts and each part is configurable according to the requirement of the administrator. Last part of VTMCI is for our actual resources; it may use as resource object to simulate algorithm and new policies or new configuration, or can implement this part according to actual resources and communicate with other cloud manager.

In next section, we describe each of these parts and define their properties and their duty in our systems.

6.1 Cloud Interface

This part of VTMCI systems is responsible for accepting new requests of system from user or any other cloud manager. Cloud interface consists of three major parts. First part is responsible for accepting request from user, second part is responsible for accepting request from other clients and the last part is responsible for sending requests to other machines. Figure 24 is representing an overview of our cloud manager and cloud interface.

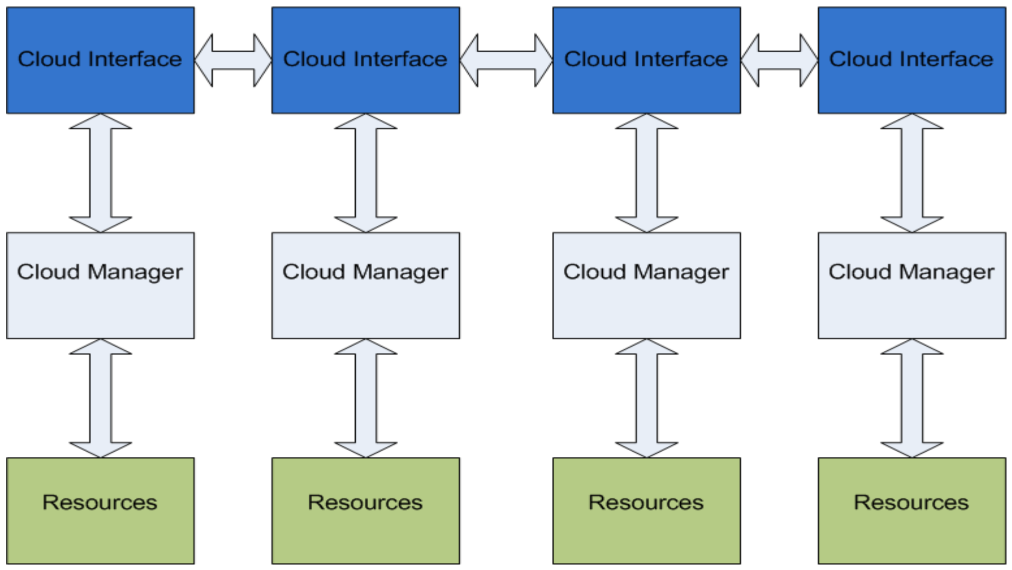


Figure 24 Combination of Cloud Managers and Cloud Interfaces

6.1.1 User Cloud Interface

Implementing first part of this cloud interface needs to define a method for user to put its request to our system. We supposed each user sends one request to our system that consists of virtual topologies that may change during the time. If one user wants to send more than one virtual topology to the system, needs to define these virtual topologies as separated requests. In each request user needs to define bunch of virtual topologies which they are changing during the time. So they need to specify their virtual topologies and then set a validation time for this topology. When one topology in one request changes, VTMCI automatically determine this change and do required actions to implement changes of the request.

In each virtual topology, user is required to fill his/her requirements. For each virtual topology there is a group of connected virtual machines. User needs to define virtual topology and connection between virtual machines in each topology. User may define one ID for each virtual machine. With this way, when change occurs in virtual topologies, we may migrate some of old virtual machines (according to their IDs). But this system will keep an internal ID for request, virtual topology and virtual machine.

For specifying virtual machines, user needs to send the required memory and CPU for each of them. For links between these virtual topologies, user needs to specify bandwidth and latency. But as we mentioned before, cloud manager can change interface and ask for more or less information from user. (For example number core of each virtual machine)

6.1.2 Inter Cloud Interface

This part of cloud interface implemented as an XML-RPC server. Each cloud interface is consists of XML-RPC server. Each request to VTMCI from other cloud interfaces will come through this server.

Once we receive a request from other cloud interfaces, this request is in a form of multiple virtual topologies that changes during time and we can accept or reject this request according to our policies. But before accepting this request, this request will send to cloud manager component to check is it possible to accept request or not. If it is possible or impossible to accept this request, cloud interface send a message to inform the remote client of the decision. In current design we supposed each cloud manager has access to all images, but cloud manager may need to modify this part according to their infrastructures. They may need to send virtual machines over network.

When VTMCI have problems for satisfying local requests, it needs to send a message to the user or send this request to the other server. Depending on pre-defined policies it needs to decide to send request to other cloud or send it back to the user. Current implementation will notify user and let him/her to decide.

If VTMCI needs to send request to the other cloud, it must make Xml file from properties of the request and put virtual topologies information in xml file and send it to the other cloud manager. But VTMCI have to keep a track of all cloud managers. With this way, it can determine which cloud is more suitable for this request.

6.2 Cloud Manager

Cloud manager is the most important part of VTMCI. Cloud manager consists of four major parts. This component is retrieving data about Virtual topologies from user interface and also information about physical resources. Cloud manager is responsible for fault in physical machines and changes in status of requests. This manger works as a coordinator between physical layer and user interface.

Cloud manager have four major component and some sub-components. Each of these components is responsible for one part of system and all of them are customizable according to the cloud administrator's requirements. Figure 24 represents these four components and relation among them.

Analyzer in system is responsible for taking information of request from cloud interface and also information about physical machines situation from monitor. VTMCI have a Mapper component, this component receives information of physical topology (physical machines and physical links between them) and description of virtual topologies. Mapper components find an optimal mapping between virtual topology and physical topology and send this mapping to resource allocator. Resource allocator is responsible to determine which virtual machine should be added to physical machine and which virtual machine needs to migrate from one physical machine to the other one. After that, resource allocator sends requests to the resources. Another important component of cloud manager is monitor. Monitor constantly monitors the situation of resources and their capability. If any change happens on physical resources or any fault occurs on one or more machines. This component will notice this error and send it to the analyzer.

In next sub-sections, we explore each of these parts and explain the duty of each part.

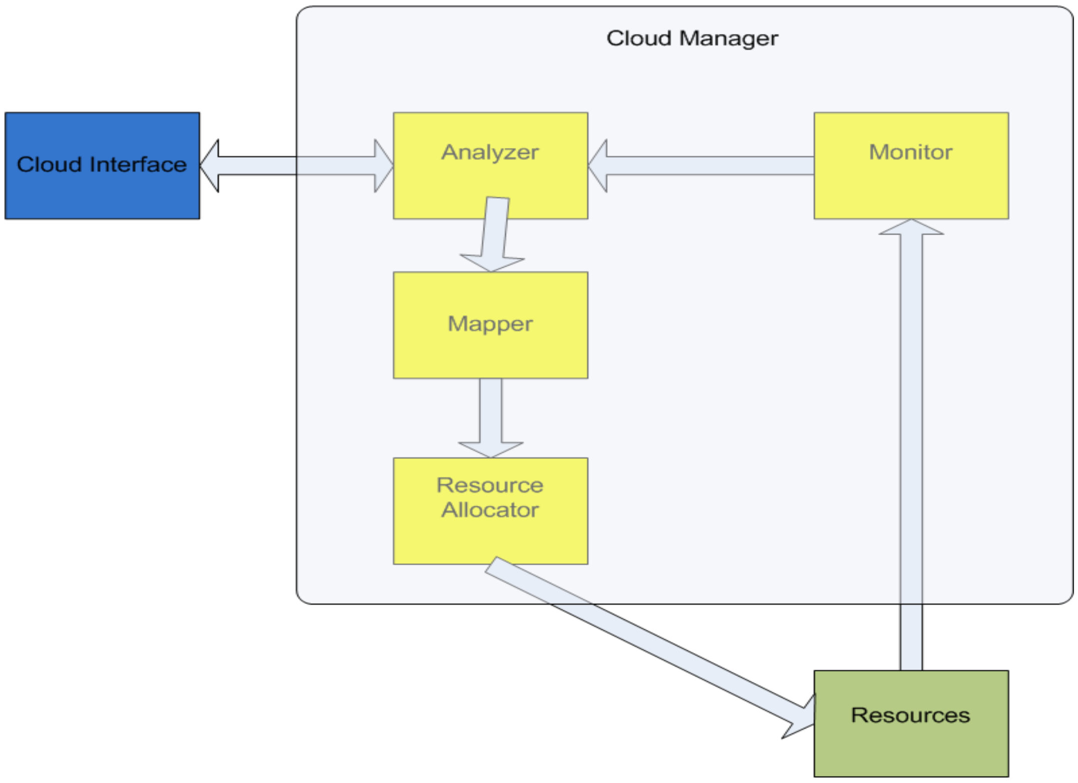


Figure 25 Cloud Manager Components

6.2.1 Analyzer

This component is most important component of cloud manager. This component receives information from cloud interface, Mapper and monitor. Then coordinate cloud manager according to this information. Analyzer receives information about new request, change in request or any change in physical resources status. According to this information analyzer must determine the best action of system. For example if a new request comes to the system and system doesn't have enough resources to accept this request, analyzer must send a feedback to the user interface.

Analyzer has two important parts. First part is responsible to check for failure of physical machines and do appropriate action according to failure of that resource and policies. Second part is in charge of changing virtual topologies of requests. If analyzer receive any change in one of the topologies, first it needs to determine is it possible to accept change or not and then it must pass it to the Mapper.

Check for failure component is responsible to get information from monitor and determine is it possible to add virtual machines to other physical resources or it must migrated to the other cloud. Follow of events is from monitor to analyzer and then back to the analyzer. First we receive a fault message from monitor and send it to the analyzer. Analyzer determines is it possible to support these virtual machines with current resources or not. If it is possible then analyzer send list of failed virtual resources to Mapper. Mapper checks are it possible to put these virtual machines on actual resources (according to network constraints). If it is not possible, Mapper sends a message to analyzer. In analyzer there are policies for this situation. Analyzer may remove some virtual topologies from physical resources to support other virtual topologies or simply send a message to the user to decide about this situation.

Changing in request component of VTMCI is supposed to support dynamic change of virtual topologies based on user request. So this component is responsible for supporting changes in virtual topologies.

If changes occurs in user's request, cloud interface send changes to the analyzer. Analyzer first compare requirement of this new virtual topology with current situation of physical resources and then if it is possible to satisfy this request, send it to the mapper. Mapper component check for mapping between this new virtual topology and physical machines and checks is it possible to map this virtual topology to physical machines or not. Then it will send messages to the analyzer and resource allocator. After that analyzer receive this response and send it back to the cloud interface. Cloud interface can decide to show this message directly to the user or sending it to the other cloud manger according to the situation.

In analyzer component system administrator is able to add more policies and more restriction. For example it can add a component to do appropriate action when change occurs in physical resources.

6.3 Mapper

This component consists of three sub components. The main reason of make a mapper in VTMCI is to find a mapping from virtual topologies to the physical machines. But cloud administrator may need different algorithms for different situations. For example when topology change occurs in one request, Mapper must check is it possible to do migration and new deployment as minimum as possible. Or in a case of physical machine failure, it may need to determine which virtual topologies needed to be removed from system.

Mapper component has three major parts. First part is for first time requests, once system receives a request; Mapper needs to find a mapping between this new virtual topology and physical topology. Second component is for next times, when request changes in the system, this request may change during time and for second time and times after that VTMCI needs to have an algorithm to find a mapping with minimum number of migration of virtual machines and virtual machine deployment. Last part of Mapper is the part is responsible for re-map those virtual machines has been removed from physical resource because of failure.Current implementation of VTMCI using greedy algorithm for all Mappers. But it is easily configurable for everyone to change these algorithms.

First time mapper is responsible for mapping virtual topologies to physical topologies for first time. When user submits one request to the system, this request consists of a list of virtual topologies that change during the time. When mapper components want to map virtual topology for first time, it doesn’t need to consider migration of virtual machines. In this case, mapper must put these set of virtual resources to physical resource according to predefined policy. This policy may define by administrator of cloud system.

Second time mapper is responsible for changing virtual topology according to the user request. This component has been designed for this situation. In this situation Mapper needs to consider history of the request.

When one virtual topology is located in system and want to change it to another virtual topology, system need to check the new topology with the old one and try to minimize number of migration and new deployment of virtual machine. In this component, it is possible to map some virtual machine to their old position and deploy the others to the new physical machines.

By the way, it is possible; this Mapper couldn’t find any feasible solution for new virtual topology. If this situation happens, Mapper algorithm needs to send a message to analyzer and inform cloud interface from the situation.

When failure occurs in physical infrastructure, monitor notified of this error and sends a message to analyzer, then analyzer send the new status of physical machines and a list of failed virtual machines to the Mapper component. In Mapper, failure time Mapper handles this problem.

Failure Mapper algorithm is responsible for mapping failed virtual resources to other physical resources. Like the other components of Mapper, it may be possible or impossible to find a feasible mapping to satisfy the requirement of failed virtual machine and their representative virtual topologies. If it is not possible to find a mapping for this set of virtual topologies, then Mapper will send a message to the analyzer and analyzer respond this message according to policy.

6.4 Monitor

This component is responsible for constantly monitoring the situation of physical resources. Any change on physical resources determined by this component and will forward to the analyzer.

System administrator can implement this component in different ways according to the communication system of physical machines. But in current implementation, monitor component has implemented by polling method. Monitor calls a function of each physical resource and ask about the current situation of this resource.

6.5 Resource Allocator

Resource allocator is responsible for allocation of virtual machines to physical resources. Figure 22 represent the overall functionality of resources allocator. Resource allocator has methods to deploy, migrate or delete one virtual machine from physical machine. It is possible for administrator of cloud to add more functionality to this component.

Default implementation of this component working with mock objects and for each kind of resources, cloud administrator must define methods to use actual resources. Another option is using these mock objects and in each of them makes a transaction to the real infrastructure.

Most reliable cloud infrastructures have API for connecting to their system and use those systems via programming languages. These systems usually using web services to communication with other programs. As shown in figure 26, it is possible to communicate with various types of cloud infrastructure managers. In next section, we describe briefly how to communicate with some of these systems.

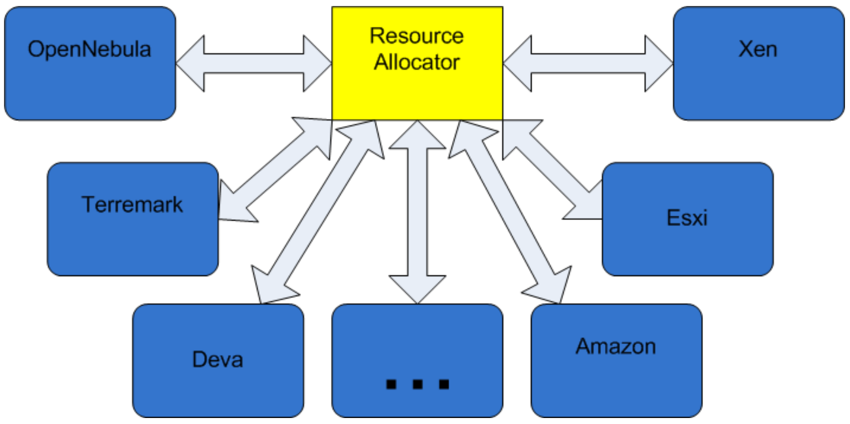


Figure 26 Resource Allocator

7. Conclusion and Future Work

This study consists of a survey of different virtualization technologies and a brief introduction to similar existing software solutions. It proposed three automotive strategies for designing and loading different networks of virtual machines on cloud infrastructure. Virtual Lab allows creating a topology of virtual machines, saving this topology and loading it as an image. Additionally, integration of Virtual Computing Lab and Open Vswitch, provides users with a powerful interface to manage all traffic among multiple virtual machines.

Virtual Lab has a simple interface that facilitates designing different topologies and saving them as a template in the provisioning module. And then, according to a user's request, VL loads appropriate VMs by using VMware Server cloud manager and establishes a network among these VMs by using software switches and routers.

Integration of Open Vswitch and Virtual Computing Lab creates a powerful solution that has features of both projects. Virtual Computing Lab allows users to reserve their appropriate images, while Open Vswitch allows users to have a software network among multiple virtual machines. The integration of these two projects allows users can reserve a network topology of virtual machines.

Several experimental validations have been used to show the correctness of this solution. In each experiment, we had measured different factors of our systems such as remote desktop protocol delay, network bandwidth, download time and ease of use. Each solution has its own pros and cons and is suitable in specific conditions.

The next phase of this software will be a complete package of installable VL for first solution and also integration of VCL and Open Vswitch as a new version of Apache VCL. Then we will integrate VL and VTMCI to create a portable software for managing a topology of virtual machines.

VL package will consist of an interface for designing network topology and scripts for determining specification of each physical machine. It will discover the hardware specification of a new host and load a sufficient number of virtual machines on each physical host. Additionally, we will add another component which allows different VLs in different hosts to communicate with each other.

New versions of VCL have an interface for specifying the number of virtual network interfaces of each VM and allow users to design a network among those VMs. It also loads one instance of Open Vswitch on each physical host and by using this software switch; it creates a network among VMs automatically.

Bibliography

[1] R. J. Creasy, "The Origin of the VM/370 Time-Sharing System," IBM Journal of Research and Development, september 1981.

[2] R. P. Goldberg, "Survey of Virtual Machine Research," IEEE Computer Magazine, 1974.

[3] [Online]. Available: http://www.VMware.com/.

[4] [Online]. Available: http://aws.amazon.com/ec2/.

[5] Paul Barham , Boris Dragovic , Keir Fraser , Steven Hand , Tim Harris , Alex Ho , Rolf Neugebauer , Ian Pratt , Andrew Warfield, Xen and the art of virtualization, Proceedings of the nineteenth ACM symposium on Operating systems principles, October 19-22, 2003, Bolton Landing, NY, USA

[6] "VMware Server," [Online]. Available: http://www.VMware.com/products/server/overview.html.

[7] "VMware® Server 2, A Risk-Free Way to Get Started with Virtualization," [Online]. Available: http://www.VMware.com/go/patents.

[8] "VMware Vcenter Server," [Online]. Available: ttp://www.VMware.com/products/vcenter-server/overview.html.

[9] "VMware ESX and VMware ESXi, The Market Leading Production-Proven Hypervisors," [Online]. Available: http://www.VMware.com/go/patents.

[10] "VMware Vsphere," [Online]. Available: http://www.VMware.com/products/datacenter-virtualization/vsphere/index.html.

[11] "VMware Vcenter," [Online]. Available: http://www.VMware.com/products/converter/ converter.

[12] "VMware ESX/ESXI," [Online]. Available: http://www.VMware.com/products/datacenter-virtualization/vsphere/distributed-switch.html.

[13] "vSphere Networking,ESXi 5.0,vCenter Server 5.0," [Online]. Available: http://www.VMware.com/support/pubs.

[14] "Open Vswitch," [Online]. Available: http://openvswitch.org/.

[15] "KVM," [Online]. Available: http://www.linux-kvm.org/page/Main_Page.

[16] "Virtual Box," [Online]. Available: https://www.virtualbox.org/.

[17] "Open Vswitch diagram," [Online]. Available: http://blog.sflow.com/2012/05/tunnels.html.

[18] "xCAT," [Online]. Available: http://xCAT.sourceforge.net/.

[19] "xCAT Architecture," [Online]. Available: http://sourceforge.net/apps/mediawiki/xCAT/index.PHP?title=XCAT_2_Architecture.

[20] "VL diagram for IT automation Class," [Online]. Available: http://ita-portal.cis.fiu.edu/.

[21] Averitt S., Bugaev M., Peeler A., Shaffer H., Sills E., Stein S., Thompson J. and Vouk M. Virtual Computing Laboratory (VCL). In Proceedings of the International Conference on the Virtual Computing Initiative. (Research Triangle Park, North Carolina, USA, May 2007).

[22] "Apache VCL," [Online]. Available: http://vcl.apache.org/.

[23] V. B. Jithesh Moothoor, "A Cloud Computing Solution for Universities: Virtual Computing Lab".

[24] "VCL academic partners," [Online]. Available: vcl.ncsu.edu/academic-partners.

[25] "VCL Architecture," [Online]. Available: https://cwiki.apache.org/VCL/vcl-architecture.html.

[26] Windows server 2008Available:http://www.microsoft.com/en-us/download/details.aspx?id=8371.

[27] Ubuntu 8.04Available:http://releases.ubuntu.com/8.04/