ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology
Volume 1, Issue 4, June 2012

Desktop virtualization using SaaS Architecture

Pranit U. Patil, Pranav S. Ambavkar, Dr.B.B.Meshram, Prof. Varshapriya
VJTI, Matunga, Mumbai, India.
pranit_patil@aol.in

Abstract **- Desktop virtualization is new desktop delivery method in which desktop operating system instance customized, build and runs in data centers and users can access application or the whole desktop by using their stateless 'thin clients' devices. This method promises significant benefits in terms of data security, total cost of ownership and manageability of large amount of operating systems instances running in corporate intranetworks. In this paper we have proposed yet Cloud architecture to provide desktop virtualization which is efficient is terms of network bandwidth and applications overhead at client side.**

Keywords **– cloud computing, desktop virtualization, virtualization. Openstack**

1. INTRODUCTION

Virtualization is relatively generalized term, which refers to the operation and management of computing elements on the virtual platform and as a resource solution in order to simplify the management and optimize resources. Desktop virtualization has varied definitions with the context in which it is implemented [1].

Desktop virtualization also referred as virtual desktop interface. Desktop virtualization can be defined as virtualization of computer desktop in order to achieve security and flexibility. The current desktop virtualization techniques propel the whole desktop on the end user machine. They pay more attention to supply high quality display effects at the client, increasing interaction between the machine and end user. This also increase required network bandwidth and processing at client side as client have to process and render the whole propelled desktop. Many times this desktop as a whole is unnecessary as end users are tend to use only applications provided within that desktop. To provide solution of this we have proposed the Yet Another Desktop Virtualization (YADV) system. This paper presents introduction to the desktop virtualization and the efficient and lightweight desktop virtualization system (YADV) using the existing available cloud stack OpenStack which will provide flexibility, extendibility and security of whole system.

2. RELATED WORK

In various fields like IT or educations there have been significant advances of the delivery of desktop environments to the end users, mainly through virtualization techniques and thin client.

There are various existing protocols and techniques are used to enable desktop virtualization. Mainly they are share the same concept of relaying the keyboard and mouse events from the client to the servers. Here server provides client only screen updates. Various efforts have been already taken the desktop virtualization products like Virtualbox, VMware Workstation which are commonly used by users does not completely realize desktop virtualization. In other words, this software runs the virtual machine instance on the local or remote computer which is possible by providing and emulates the set of hardware so that it can be transferred to other machine by means of removable media or network.

SunRay, the thin client solution from Sun Microsystems now Oracle, provides a client and server solution with token based smartcards allowing users to move from one thin client device to another and having their session managed by tracking the token ID registered on smartcard, SunRay is widely deployed within Sun and has a growing commercial customer base. It utilizes Xen and PowerVM virtualized linux operating system images[2].

Also VMware workstation based virtual laboratories which provide VNC access to virtual machines running on host machines, also there is another solution provided which uses VMware solution based on RDP rather than VNC. Adams

All Rights Reserved © 2012 IJARCET

extends the virtual laboratory through the use of central NFS storage for virtual machines. Civic provides middleware infrastructure on top of raw hardware resources to provide hosting environment for virtual machine instance and virtual machine instance and virtual network instance[3]. User can interact with the instance just like with physical machine.

However systems mentioned above all do not pay much attention on scalability, security and decreasing desktop granularity and optimize the protocol utilized between clients and servers which are important in various environments, such as, wireless networks.

3. SYSTEM MODEL

In order to make high utilization of resources and provide more secure and feasible services for users, we propose a lightweight desktop virtualization system using OpenStack cloud stack. We first describe the main framework of the system under study.

3.1. System architecture

The main part of 'YADV' is OpenStack, which is a group of open source projects, that together aim at creating and managing public as well as private clouds. Following OpenStack[4] architectural diagram gives an overall idea of cloud operations in general;

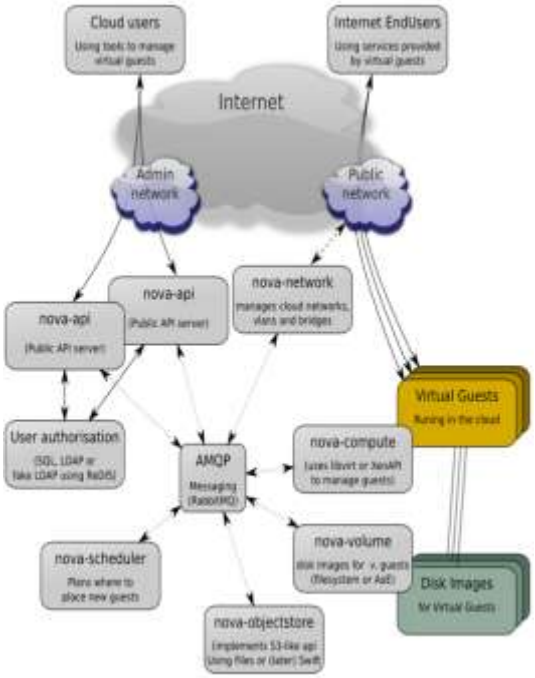

Figure 1. OpenStack Service Architecture

OpenStack helps to create Infrastructure-as-a-Service in a convenient manner by providing generic API's which can work with different cloud vendors and thus prevent vendor lock-in issues. OpenStack is not a hypervisor but one of its components controls the hypervisor and other underlying functions. Due to the scope of the paper, we will not discuss details of OpenStack and will restrict our discussion to the critical issues in VM operations.

Other components are typical cloud components like object storage, network controller, compute manager and so on. Specifically, the main management module in OpenStack is named as "nova" and thus the relative names of other components. Customers typically consume and interact with cloud services through the API's as shown above. In the following section, we briefly describe the basic VM operations and then describe our observations. For clarity we would like to emphasize that Virtual Cloud Controller or VCC is the combination of the main compute module, cloud management API's and the system administrator that operates on these two entities i.e. VCC = nova-compute module + cloud management API + system administrator). Virtual Machine Operations3.1.1 Creation: In this phase, a cloud user gets authenticated via 'user-authorization' module. It is single-sign on authorization where cloud user can access all the associated services from various components of openstack. To create an instance, a cloud user sends request to VCC along with a predefined or customized VM template that contains basic OS parameters, RAM size etc. Here, nova-scheduler picks up a computer node from pool of available resources depending upon its scheduling algorithm and then the execution phase of the instance starts on that computer node (VMM).

3.1.2 Execution: VM starts executing an instance by using customized OS templates provided by the user or by fetching an available VM image stored in Image Store(Swift). After starting an instance, a key pair is generated and gets attached to it and nova-network component assigns public IP addresses from available pool of addresses .Finally, this instance is provided to internet users along with the required information about the instance.

All Rights Reserved © 2012 IJARCET

3.1.3 Migration: It consists of moving a running instance between different physical servers (VMM) without disconnecting the client. It includes migration of memory, storage and network connectivity of a virtual machine.

3.1.4 Termination: In this phase, a cloud user or internet user sends request to terminate the running instance. Termination process consists of stopping the instance, saving user data and removing any temporary data if used by virtual machine.

Here, the following figure shows, the environment variable setup for accessing various services provided by the OpenStack

```
root@nova-controller:/usr/share/nmap/scripts# env |
NOVA_PROJECT_ID=ten
NOVA_REGION_NAME=nova
NOVA_VERSION=1.1
NOVA_USERNAME=chuck
NOVA_API_KEY=pass
NOVA_URL=http://127.0.0.1:5000/v2.0/
NOVA_AUTH_STRATEGY=keystone
EC2_SECRET_KEY=pass
EC2_URL=http://127.0.0.1:80/services/Cloud
EC2_ACCESS_KEY=chuck
OS_AUTH_USER=chuck
OS_AUTH_STRATEGY=keystone
OS_AUTH_URL=http://127.0.0.1:5000/v2.0/
OS_AUTH_TENANT=ten
OS_AUTH_KEY=pass
AUTH_TOKEN=78c246d2-437a-4bca-aacd-45aa5d5fbc89
root@nova-controller:/usr/share/nmap/scripts#
```

Figure 2. Environment Variable for OpenStack

3.2 Customization of Operating System disk images:

In YADV we are going to make customized operating system disk images to make sure virtual machine instances will run smoothly and will have all the necessary packages in it to support the connectivity to the client. As we are going to use Ubuntu Linux, the following procedure will be more or less same to the other Linux based distros. Customization of operating system involves localize to certain language, install or remove a software application, update software and change the system defaults [5]. Here we'll install SSH server and required applications which are required at client side.

The example shown here uses the ubuntu-9.04-desktop-i386.iso Desktop CD.

```
mv ubuntu-9.04-desktop-i386.iso ~/livecdtmp
```

Move or copy it into an empty directory

```
mkdir ~/livecdtmp
mv ubuntu-9.04-desktop-i386.iso ~/livecdtmp
cd ~/livecdtmp
```

Mount the Desktop .iso

```
mkdir mnt
sudo mount -o loop ubuntu-9.04-desktop-i386.iso mnt
```

Extract .iso contents into dir 'extract-cd'

```
mkdir extract-cd
sudo rsync --exclude=/casper/filesystem.squashfs -a mnt/ extract-cd
```

Extract the SquashFS filesystem

```
sudo unsquashfs mnt/casper/filesystem.squashfs
sudo mv squashfs-root edit
```

To use the network connection within chroot

```
sudo cp /etc/resolv.conf edit/etc/
```

Customization

```
apt-get update
apt-get install openssh-server
apt-get install bluefish
apt-get install gnome-panel
```

Here in customization, we have installed SSH server on cooking Ubuntu desktop system which will provide client application to access the virtualized instance of the same.

Producing the CD image

```
sudo mkisofs -D -r -V "$IMAGE_NAME" -cache-inodes -J -l -b isolinux/isolinux.bin -c isolinux/boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table -o ../ubuntu-9.04.1-desktop-i386-custom.iso .
```


All Rights Reserved © 2012 IJARCET

This will produce iso image of customized desktop operating system which can we used by our YADV system which uses Openstack as its backbone. The following snapshot shows registered iso images with OpenStack

Figure 3. Registered iso images with OpenStack

3.2 Software for clients:

The software for client is lightweight and easy to use, while showing high performance and high security. It has two main functions, one is to receive encrypted data from servers and to process and display them on screen; the other is to intercept input event from local machine and send them to server. Actions like window resizing, configuration, themes and other local graphical properties also be detected by the software at client side.

3.3 Interaction between two sides

In this paper, we use existing SSH protocol to implement interaction between client and servers SSH are highly secure method of communication which provided encryption to prevent data interception. This protocol simplifies the work of data processing significantly and makes the system have good portability among different platforms including Windows.

4. THE YET ANOTHER DESKTOP VIRTUALIZATION SYSTEM

In this section, we present the design of the proposed desktop virtualization system.

4.1 Implementation of application streaming

Many systems like VNC, Windows based RDP or Teamviewer only provides full screen display service. However, most of the times, user wants to concentrate on single application but not the whole desktop, to that these systems does not make any sense. In this paper, the following issues are studied and implemented: How to obtain information within single window at client side? How to propel software list and not all the desktop to the client? Which client side application can be used to ease the burden of complex configuration?

In this system, we used simple technique to deal with above problems. Our prototype is based on Linux servers and Linux or Windows clients. The X-window system is a graphic user environment at server side. In the system we are going to use X over SSH to get work done along with 'gnome-terminal'[6] software application which will propel only software list or panel to the client, thus background information about the rest desktop improving the performance of virtualized instance.

The figure 4 shows the propelled software list on the client machine.

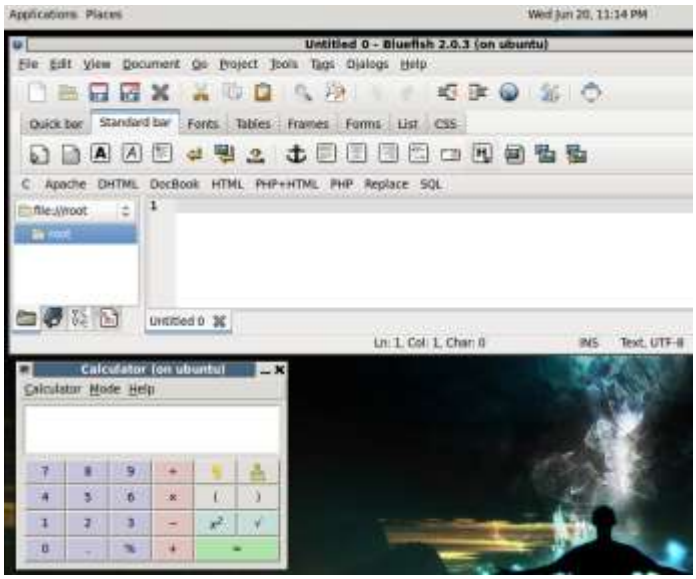

Figure 4. Propelled software list on client machine

4.2 Combination of UI elements

All Rights Reserved © 2012 IJARCET

With the assistance of 'gnome-panel' software application, the pixel data on server side can be shown at client side. 'gnome-panel' does provide flexibility to put itself on any edge of the current desktop screen of client. Also if the client is also linux, then it shown new propelled software list on the existing software panel with hover effect.

4.3 Client side application and implementation

To use desktop propelled with YADV, client needs to have very simple and lightweight existing softwares applications like putty or ssh client.

If the YADV client is linux machine, then user needs to issue only following commands

```
ssh –X ipaddress_of_YADV_server
```

If the YADV client is windows, it needs to install Xming X-server [7] which will acts as a client and sends keyboard and mouse events to the YADV server.

5. CONCLUSION AND FUTURE WORK

In this paper, we have proposed YADV, a novel approach to provide remote virtualization services to the end user,. In the proposed framework, server resources are integrated into powerful and scalable computing cloud stack i.e. OpenStack which used to make more efficient use of available hardware. Moreover all the applications are running on virtual machines and monitored using OpenStack only. The simulation studied conducted on this YADV shown us that it provides better QoS and efficiently allocates resources among users.

Future works includes the system development based on Web 2.0 which will eliminate need of using any client side application expect browser. More work is also needed to further optimize the customized disk image to provide quick and easy customization according to end user need.

REFERENCES

1) Desktop Virtualization Technologies and Implementation: Pranit Patil-IOSRJEN

2) http://www.oracle.com/us/technologies/virtualization/061984.html

3)http://pdf.aminer.org/000/299/287/specification_based_computing_environments_for_information_management.pdf

4) http://openstack.org/

5)https://help.ubuntu.com/community/LiveCDCustomization

6) http://library.gnome.org/users/gnome - terminal

7) http://sourceforge.net/projects/xming/

All Rights Reserved © 2012 IJARCET