DESKTOP VIRTUALIZATION

A PROJECT REPORT

Submitted by

S. EUGIN RAPHAEL **09CS14**

V. VENKATESH **09SCS28**

R. VIVEK KANNAN **09SCS30**

In partial fulfillment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

COIMBATORE INSTITUTE OF TECHNOLOGY

(Govt. Aided Autonomous Institution Affiliated to Anna University)

(Accredited by National Board of Accreditation Council)

COIMBATORE - 641014.

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2013

COIMBATORE INSTITUTE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University)

COIMBATORE-641014

BONAFIDE CERTIFICATE

Certified that this project report "**DESKTOP VIRTUALIZATION**" is a bonafide work of

Eugin Raphael S	**09CS14**
Venkatesh V	**09SCS28**
Vivek Kannan R	**09SCS30**

In partial fulfillment for the award of the Degree of Bachelor of Engineering in Computer Science and Engineering of Anna University, Chennai during the academic year 2012-2013 who carried out the project work under my supervision.

Prof. K.S.Palanisamy, M.E

HEAD OF THE DEPARTMENT,

Computer Science and Engineering,

Coimbatore Institute of Technology,

Coimbatore - 641014.

Mrs. D.Mohana, M.E

PROJECT GUIDE,

Computer Science and Engineering,

Coimbatore Institute of Technology,

Coimbatore - 641014.

Certified that the candidates were examined by us in the project work viva-voce examination held on ……

Internal Examiner **External Examiner**

Place: Coimbatore

Date:

ACKNOWLEDGEMENT

We express our sincere thanks to our Secretary **Dr.R.Prabhakar** and our Principal **Dr.V.Selladurai** for providing us a greater opportunity to carry out our work. The following words are rather very meagre to express our gratitude to them. This work is the outcome of their inspiration and product of plethora of their knowledge and rich experience.

We record the deep sense of gratefulness to **Professor K.S.Palanisamy**, Head of the department of Computer Science and Engineering, for his encouragement during this tenure.

We equally tender our sincere thankfulness to **Mrs. D.Mohana**, Associate Professor and our supervisor, Department of Computer Science and Engineering, for her valuable suggestions and guidance during this course.

During the entire period of study, the entire staff members of the Department of Computer Science and Engineering have offered ungrudging help. It is also a great pleasure to acknowledge the unfailing help we have received from our friends.

It is a matter of great pleasure to thank our parents and family members for their constant support and co-operation in the pursuit of this endeavour.

ABSTRACT

Virtualization emerged as a practice to optimize the use of expensive computing hardware. Desktop virtualization targets to reduce the complexity associated with deployment and maintenance of client devices, which ultimately helps to reduce desktop management costs. In Server-based Desktop Virtualization, the users physically work on thin clients, but the operational environment that they interact with is actually running on a remote Server. The user input is transmitted across the network to the remote server, and the user interface i.e., the virtual desktop is presented back through the network to the end-user. We propose a server based desktop virtualization system, in which mobile phones can be used as thin clients and desktop computer acts as the server. All the processing is done centrally on the server itself. Nothing is executed or persistent at the client side.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

1. INTRODUCTION

A type of virtualization which is actually rapidly growing faster than either server or OS virtualization, both in market and in importance is desktop virtualization [3]. The primitive concept of desktop virtualization is based on application execution from server to remote client. Virtualization extracts the physical physiognomies of computing from the way in which other systems, applications, or end-users interact with those resources. It permits individuals to use computing competences from a solo application to a complete operating system without being tangled to the explicit physical hardware or other resources that are supporting to those capabilities.

Server-centered desktop virtualization [5] abstracts physical desktops from how end-users work together with them. The users actually works with their thin client, but the computing environment that they interact with is actually running on a single system which may be a remote one typically a data center server. Any user input (keystrokes, mouse clicks, etc.) is sent across the network to that system, and the user interface is presented back across the network to the end-user. In server virtualization for example, a layer known as the *hypervisor* allows multiple virtual computers to run on a single physical machine – enabling a level of flexibility that many organizations are appreciating today.

Such ideas have been broadened to encompass virtualization technologies for desktop computing. In traditional desktop models, the computer runs an operating system where individual applications are executed with their user interface displayed on the computer screen. By introducing virtualization however, we break the direct connection between physical hardware, operating system, application and display between layers.

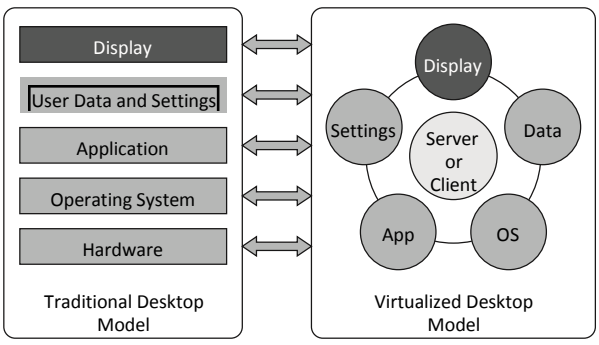

Fig.1 Desktop virtualization breaks traditional bindings

Unlike client-server computing [4], application virtualization and application streaming which focus on delivering individual applications desktop virtualization provides a complete operating environment, including the operating system, applications, and data. In addition, unlike systems that provide virtualization on desktop systems, server-based desktop virtualization removes the dependency on specific local PC configurations.

DESKTOP VIRTUALIZATION APPROACHES

Desktop Virtualization offers a number of options to deliver desktop capabilities to users. What they all share is that they're different from the traditional desktop model. The desktop is the most common point of access for users and, as such, how well it performs has a disproportionately high bearing on user productivity, as well as satisfaction and perception of IT.

USER STATE VIRTUALIZATION

The simplest option is to virtualize the link between a user's configuration settings and/or data, storing this information on a server so that any connected desktop client can access a user-session. User state virtualization works well in tandem with application virtualization and standardized desktop environments, where each desktop runs a similar set of applications. From the user perspective, not only can users log on where they want, but also, if something goes wrong with the desktop, they can continue working on another machine with minimal interruption.

APPLICATION VIRTUALIZATION

Another approach to desktop virtualization is for individuals to access applications that have been summarized and streamed, in whole or in part, down to their local computing device. Once the individual has completed the task, the application could be automatically removed from the local machine and made available for reuse on another machine or it could remain on the local machine. Streaming applications often require broadband network connections.

In desktop virtualization, processing is done using powerful and well-equipped servers in the data center and only the image or the user interface is sent to the desktop or thin client computers. Another way of virtualizing desktops is by sharing or distributing the processor/memory of a single desktop to create multiple desktops by just adding monitors,

keyboard, mouse and low cost access devices for each user. The architecture for creating such shared desktops is shown in Figure 2. Each virtualized desktop has an access unit which can be used instead of CPU that connects to the network switch using RJ-45, on one side and keyboard/mouse/monitor using common interfaces like PS/2, USB, and VGA on the other side. On the right hand side, the network switch connects to a host computer.

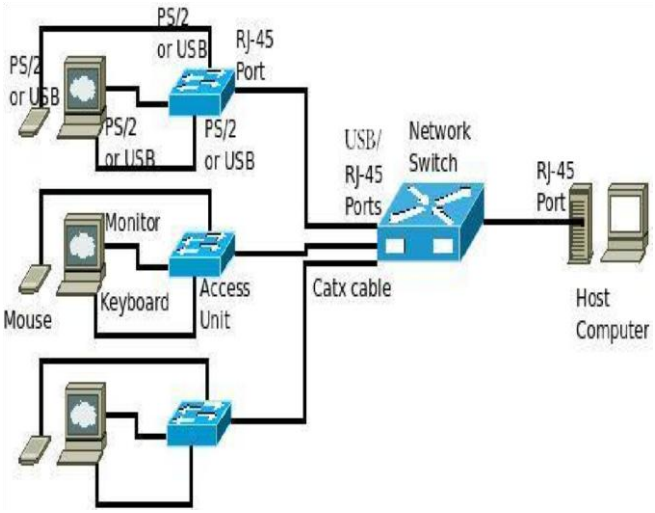

Fig.2 Working of Desktop Virtualization

Basically this works on the principle that the desktop computers available today with multi-core processors/ high capacity of RAM etc., are quite powerful and its huge amount of processor/ memory capacity is never completely utilized by a single desktop. So, that excess capacity can be shared – either with one more desktop user/ few more desktop users – depending on the load/ type of applications.

SESSION VIRTUALIZATION

Session virtualization is derived from the 'thin client architecture' model, popularized by Citrix in the 1990s. In this configuration, multiple desktop clients log onto a single instance of an operating system running on a single server. In essence, all applications or the entire desktop runs on the same server, and the display graphics are transmitted to a desktop client. The client can be 'thin' given that it only has to be able to display graphics. Indeed, it may not need any processing power at all: 'ultra-thin clients' contain little more than a network port and a graphics card.

VIRTUAL DESKTOP INFRASTRUCTURE (VDI)

VDI is an extension to the thin client idea. In this case however, as well as using one server to service the processing needs of individual users, each user has an operating system instance on the server, running as a virtual machine. VDI is often termed as ‘desktop virtualization’ in general.

BLADE-BASED VIRTUAL DESKTOPS

‘Blades’ are standardized server computers designed to slide into a specially designed rack to minimize space and power requirements. Like session virtualization or VDI, the blade-based virtualization model means that the processing is done on a server and the display is transmitted to the user’s desktop. In this case however, each blade runs a single operating system which is allocated to an individual user at runtime.

SINGLE-DESKTOP VIRTUAL MACHINE

For completeness, we need to give a mention to singleinstance desktop virtualization, where virtualization software is run on a desktop computer. The model was popularized by developers who needed an operating system for testing but didn’t have spare hardware to run the second instance, as well as IT pre-sales engineers who could run up a version of their software in a virtual machine.

Importantly, single-instance desktop virtualization doesn’t require a network connection because the virtualization software runs on the same device as the desktop itself. These days, single-desktop virtualization is also incorporated into Windows 7 in the guise of XP Mode enabling legacy Windows XP applications; it’s also a popular mechanism for running Microsoft Windows on the Apple Macintosh.

BENEFITS OF DESKTOP VIRTUALIZATION

2. BENEFITS OF DESKTOP VIRTUALIZATION

The primary reason why we might want to under-take a desktop virtualization initiative is to centralize the management of our users' desktops. While specifics depend on the selected model, the essential driver behind desktop virtualization in all of its forms is to enable user environments to be controlled and managed from a central point. This can simplify operational challenges.

- Flexibility
- Security and
- Availability

Breaking the bond between physical technologies and the applications that run on them, desktop virtualization can mean more flexibility for users. Depending on the selected virtualization options, users can be free to access applications and data wherever it's most convenient to them, all the while taking their own configuration needs into account. Accessing data and/or applications from whichever computer is available, for example including running the work environment on a home PC. Running an application with specific configuration requirements, in parallel with other applications and operating systems without conflict between them. Provisioning and allocating applications more flexibly from a central point.

BENEFITS OF USER STATE VIRTUALIZATION

This model is the simplest of all virtualization options, given that it builds on top of the traditional file server and desktop client model and so requires little in the way of new hardware or software. As well as the flexibility benefits from being able to log into any appropriately configured desktop computer, this model also affords availability benefits in that user data and state information are stored on a central server [6].

BENEFITS OF APPLICATION VIRTUALIZATION

The main benefit of application virtualization is user flexibility, because applications can be linked to users rather than specific computer hardware. It also offers application stability and reliability, given that every application has its own protected configuration. One can build and deploy cleaner operating system images and reduce the time they spend in application interoperability testing, application deployment and patch management. Application virtualization can also be used to reduce licensing costs by providing access to certain applications only when needed [6].

BENEFITS OF SESSION VIRTUALIZATION

The main benefit of the session virtualization model is centralization of control and data storage, though it can be made quite flexible by offering multiple configurations to meet the needs of individual user categories. However, like virtual desktop infrastructure and blade based virtual desktops, the session virtualization model depends on the network being available and so is more suited to corporate environments [6].

Option Type	Flexibility	Control	Security	Availability	Offline Access	Performance
User state virtualization	◖	◖	○	●	●	●
Application virtualization	●	◖	◖	◖	●	●
Session virtualization	◖	●	◖	●	○	◖
Virtual desktop infrastructure	◖	●	◖	●	○	◖
Blade-based virtual desktops	◖	◖	◖	◖	○	●
Single-desktop virtual machine	●	◖	◖	◖	●	◖
Key: ● Fully applicable ◖ Partially applicable ○ Not applicable						

BENEFITS OF VIRTUAL DESKTOP INFRASTRUCTURE (VDI)

As with session virtualization, the main benefit of VDI is centralized control, and equally the model has considerable flexibility when it comes to offering different options for different user categories. Given the reliance on virtual machines, this model also has availability benefits because a virtual machine can more easily be copied or moved than a physical machine. However, given that each user is allocated a virtual machine, VDI requires more server and storage resources than session virtualization.

BENEFITS OF BLADE-BASED VIRTUAL DESKTOPS

While benefits of the blade-based model are across the board, this model is particularly applicable to heavy processing and graphics applications such as computer-aided design. This model also takes advantage of the space- and power-saving benefits of blades, as compared to running individual servers. However, the one-blade-per-user model is expensive in hardware terms when compared to deploying session virtualization or the VDI model.

BENEFITS OF SINGLE-DESKTOP VIRTUAL MACHINE

The single-desktop model offers flexibility to individual users, particularly where it's necessary to separate business from non-core working; for example, to run a business environment on personal equipment, or to keep multiple client environments separate. This is also a popular approach for software developers to, say, create target computer environments for test purposes. This model also has management benefits, in that the virtual machine can be controlled and distributed centrally, for example to preconfigure a virtual machine to assure compatibility with a legacy application.

A DESKTOP VIRTUALIZATION SYSTEM

3. A DESKTOP VIRTUALIZATION SYSTEM

The proposed System comprises of devices with browser and the PC as a Server. The Devices may use technologies like Wi-Fi to communicate with the Server. The general mechanism of the system can be explained as the device will send the request to the server to execute a particular application that would otherwise not run on the device.

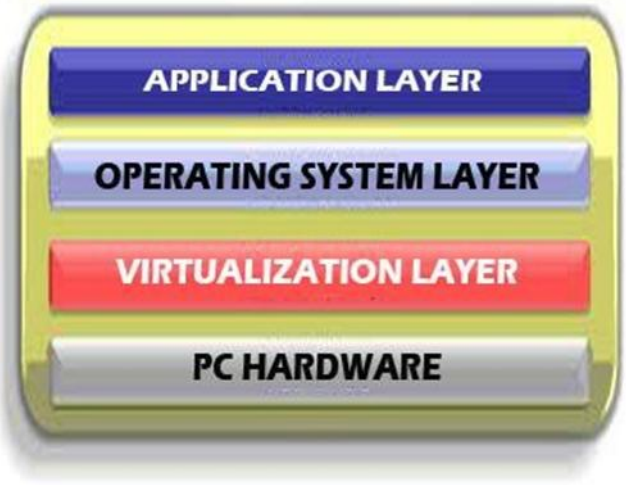

Fig.3 Layers of Desktop Virtualization

The server will respond by executing that application at its own end and only sending the UI to the device. At the client-side, the user may give any input, which is transmitted back to the server. The server will process the user input and send the resulting UI back to the client. In this manner, the proposed system will help clients to also execute the applications that otherwise would be impossible to execute on their devices.

SYSTEM ARCHITECTURE

The proposed System comprises of four layers as shown in Figure 3.

PC HARDWARE

When this approach is applied to desktops, it is clear that the flexible computing components that form the basis of flexible desktops and the user's operating environment can be dynamically delivered to any end-user device, maintaining a consistent delivery experience regardless of endpoint scenario. Conversely, a purpose-built desktop can be delivered only with the applications, settings, and services currently required by the user.

VIRTUALIZATION LAYER

With purpose-built desktops, if a user needs for a session access to a compiler, they do not need to wait for the desktop indexing service to initialize. When combined with a centralized application and data infrastructure, resources are conserved by delivering only the applications that are required.

APPLICATION LAYER

Application Delivery can also spread or mitigate the risks inherent in complex environments where applications coexist. Overall availability can be increased by spreading services across the entire logical datacenter, or in response to an individual outage.

OPERATING SYSTEM LAYER

More efficient utilization of hardware and other resources can be achieved when resources are not dedicated to a specific service. Over-utilized services can be distributed to be shared by under-utilized platforms.

This process of decoupling is so fundamental to best practice application delivery that there are very few exception cases. Exceptions may exist when a load is fixed and predictable, or when privacy or other legal issues arise; or if the layering created by decoupling creates an overhead that reduces the benefits significantly. The overall system architecture is shown in figure 3.

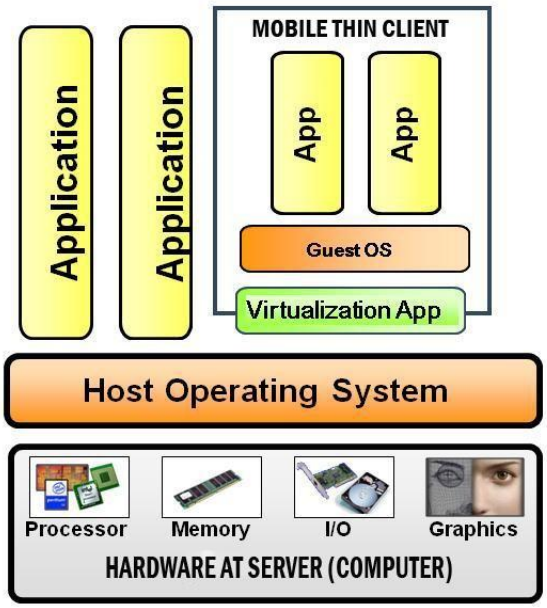

Fig.4 System Architecture

CLIENT REQUIREMENTS

Just a browser enabled device with flash support providing users with the flexibility to work anywhere and everywhere. Device should have some means for connectivity like Wi-Fi or Ethernet.

SERVER REQUIREMENTS

The server must meet the following requirements. The same machines can also be used to provide primary and secondary storage, such as via local disk or NFS.

- Ubuntu 12.04 LTS
- 64-bit x86 CPU
- 2 GB of memory
- 20 GB of local disk space
- At least 1 NIC or Wi-Fi connectivity
- Statically allocated IP address
- Fully qualified domain name as returned by the hostname command

NETWORK REQUIREMENTS

Different virtual desktop interface technologies have different traffic patterns and network bandwidth requirements. In general, server-based desktop and virtualization have longterm variable bit rate traffic patterns, mainly because of the characteristics of the remote desktop protocols, which use compression techniques. The maximum bandwidth requirement can vary greatly depending on user activities, which directly affect the complexity of the desktop display. For video applications, server-side image rendering can greatly increase the bandwidth requirement to several hundred Mbps. For client-based desktop virtualization or application streams, bursty bulk data transfers may occur when the OS and applications are streamed to the client. Given these bandwidth considerations, ample network bandwidth should be allocated to handle various scenarios. Ethernet 10/100/1000-Mbps host-facing connectivity would provide bandwidth headroom. Correspondingly, bandwidth headroom should be planned in the campus distribution and core layers.

E-LEARNING

4. E-LEARNING

The ultimate objective of the project is to provide complete E-Learning solution i.e Student should be able to use whatever operating system or applications that they want to use in their minimal device. So here we are providing a desktop virtualization solution that includes

- Moodle for Course and Test management
- Gcc compiler
- Remote Terminal instead of telnet/ssh
- Control panel for the Virtual Machines
- Ubuntu 12.04 Desktop
- Windows XP Desktop

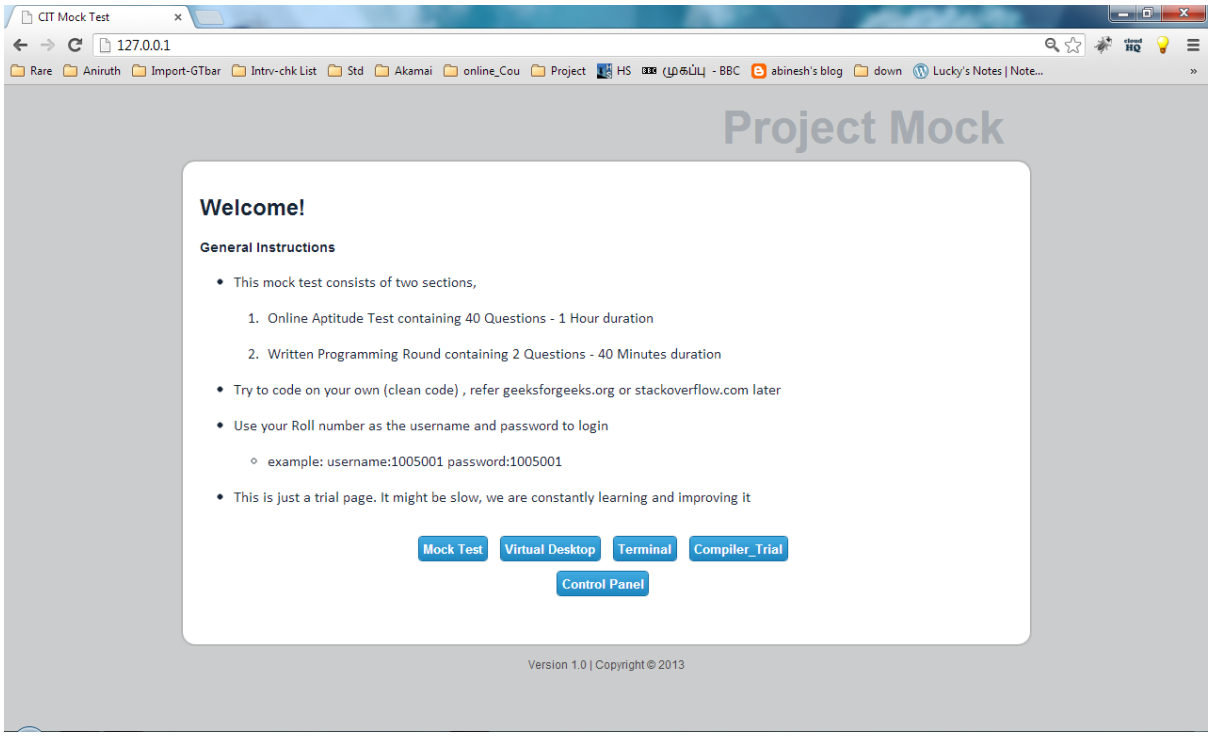

Fig.5 Project Front-page

CONFIGURING THE SYSTEM

5. CONFIGURING THE SYSTEM

This project is single server Desktop Virtualization solution using Virtual-Box as hypervisor, we need 64 bit operating system for effective virtualization solution. So, we go for Ubuntu 12.04 64 bit OS. We make use of PHP for web front end and Perl for some critical file and CGI operations, MYSQL for database. Steps that are to be followed to configure the server,

- Install Ubuntu server 12.04 64 bit OS
    - Set IP address
    - Set Domain name
    - Configure open SSH Server
- Install Apache 2
- Install PHP 5
- Install MYSQL 5.5
    - Create MYSQL database
    - Set MYSQL root password
    - Create MYSQL user
- Install Virtualbox 4.2 hypervisor
    - Configure “vboxwebsrv”
    - Configure Apache 2 with virtualbox api
    - Write scripts to manage virtual machines – create, start, stop, modify
- Install and configure virtualbox perl and php wsdl toolkit
    - Integrate RDP viewer to access the virtual machine using rdpwebcontrol
- Integrate Moodle
- Integrate GCC compiler
- Integrate Terminal client

HYPERVISOR

6. HYPERVISOR

In computing, a **hypervisor** or **virtual machine monitor** (**VMM**) is a piece of computer software, firmware or hardware that creates and runs virtual machines. A computer on which a hypervisor is running one or more virtual machines is defined as a host machine. Each virtual machine is called a guest machine. The hypervisor presents the guest operating systems with a virtual operating platform and manages the execution of the guest operating systems. Multiple instances of a variety of operating systems may share the virtualized hardware resources.

CLASSIFICATION

Type 1 (or native, bare metal) hypervisors run directly on the host's hardware to control the hardware and to manage guest operating systems. A guest operating system thus runs on another level above the hypervisor.

This model represents the classic implementation of virtual machine architectures; the original hypervisors were the test tool, SIMMON, and CP/CMS, both developed at IBM in the 1960s. CP/CMS was the ancestor of IBM's z/VM. Modern equivalents of this are Oracle VM Server for SPARC, Oracle VM Server for x86, the Citrix XenServer, VMware ESX/ESXi, KVM, and Microsoft Hyper-V hypervisor. It runs directly on the hardware

Type 2 (or hosted) hypervisors run within a conventional operating system environment. With the hypervisor layer as a distinct second software level, guest operating systems run at the third level above the hardware. BHyVe, VMware Workstation and VirtualBox are examples of Type 2 hypervisors. It runs on another operating system, such as FreeBSD, Linux, or Windows.

VIRTUAL BOX

VirtualBox is a Type 2 hypervisor. It offers extensive flexibility to meet various virtualization needs for a spectrum of users [1]. Individual users can download VirtualBox at no charge for personal use to create a powerful multiplatform desktop where almost any x86 operating system including OpenSolaris, the Solaris OS, Linux, Windows, and Mac OS can coexist . For example, VirtualBox provides a compelling test and development environment for application developers and eliminates the expense and management overhead of additional physical platforms. It offers some distinctive functionality over competitive desktop virtualization products.

VirtualBox also features a comprehensive toolbox that includes CLI and API capabilities [2]. These tools make it easy for innovative developers to embed virtualization functionality into third-party products. Developers can create alternate front-ends for the VirtualBox, or programmatically create and manage virtual machines. To deploy a precise execution environment, some manufacturers are using VirtualBox to distribute a patched OS, application libraries, and required device drivers. VirtualBox provides the tool set and the flexibility needed so that developers can easily integrate virtualization's potential into their own products.

We need to install Oracle VM VirtualBox 4.2 on our server Ubuntu 12.04 64 bit OS, Then we can simultaneously run multiple guest operating systems inside Oracle VM VirtualBox using multiple virtual machines (VMs).

The host OS can be Microsoft Windows, Mac OS, Linux, or Oracle Solaris. Many operating systems are supported as guests [1] For example, we can run Windows and Linux as guests on our Mac, we can run Windows Server 2008 and Oracle Solaris as guests on our Linux server, we can run Linux as a guest on our Windows PC, and so on, all alongside our existing applications. we can install and run as many virtual machines as we like the only practical limits are disk space and memory[1].

Oracle VM VirtualBox consists of three parts:

- The first part is the base software package that is available for each supported host operating system.
- The second part is guest additions, which can be installed on the installed guests and add support for shared folders, seamless window integration, and 3D.
- The third part is extension packs, which can be installed to extend Oracle VM VirtualBox functionality. The Oracle-provided extension pack provides support for USB 2.0, the Oracle VM VirtualBox Remote Desktop Protocol (VRDP), and the Preboot eXecution Environment (PXE) boot ROM.

  Data can be transferred to and from the guests through external iSCSI storage, with shared folders from the host or via network services. There are several options for connecting Oracle VM VirtualBox guests with the host or the outside world:

- Network address translation (NAT) networking: For clients on an Oracle VM VirtualBox private LAN, to connect them to the host's external network
- Bridged networking: Bridges guests to the host network and makes them full network citizens
- Internal networking: Binds guests to an isolated network, which is independent and separate from the host
- Host-only networking: A hybrid between bridged and internal networking, which connects the isolated private network with the host

In addition, there are several interfaces for Oracle VM VirtualBox, such as a command-line interface (VBoxManage) [2] . These interface are very flexible and enable administration of Oracle VM VirtualBox.

Fig.6 Virtual Box running Windows XP on Ubuntu OS

VIRTUAL MACHINE COMMAND LINE INTERFACE

7. VIRTUAL MACHINE COMMAND LINE INTERFACE

VBoxManage is the command-line interface to VirtualBox [2]. With it, we can completely control VirtualBox from the command line of our host operating system. VBoxManage supports all the features that the graphical user interface gives we access to, but it supports a lot more than that. It exposes really all the features of the virtualization engine, even those that cannot be accessed from the GUI.

We will need to use the command line if we want to

- Use a different user interface than the main GUI (for example, VBoxSDL or the VBoxHeadless);
- Control some of the more advanced and experimental configuration settings for a VM.

- We can specify the VM name, as it is shown in the VirtualBox GUI. Note that if that name contains spaces, then we must enclose the entire name in double quotes (as it is always required with command line arguments that contain spaces).

```
VBoxManage startvm "Windows XP"
```

- We can specify the UUID, which is the internal unique identifier that VirtualBox uses to refer to the virtual machine. Assuming that the aforementioned VM called “Windows XP” has the UUID shown below, the following command [1] has the same effect as the previous:

```
VBoxManage startvm 670e746d- abea-4ba6-ad02-2a3b043810a 5
```

- We can type VBoxManage list vms to have all currently registered VMs listed with all their settings, including their respective names and UUIDs.

Some typical examples of how to control VirtualBox from the command line are listed below:

- To create a new virtual machine from the command line and immediately register it with VirtualBox, use VBoxManage createvm with the --register option

```
$ VBoxManage createvm --name "UBUNTU" -- register
```

Virtual machine ’ubuntu’ is created.

UUID: c89fc351-8ec6-4f02-a048-57f4d25288e5

Settings file: ’/home/username/.VirtualBox/Machines/UBUNTU/UBUNTU.xml’

As can be seen from the above output, a new virtual machine has been created with a new UUID and a new XML settings file.

- To change settings while a VM is powered off, use VBoxManage modifyvm,

```
VBoxManage modifyvm "Windows XP" --memory "512MB"
```

VBOXMANAGE CREATEVM

This command creates a new XML virtual machine definition file [6].

The --name <name> parameter is required and must specify the name of the machine. Since this name is used by default as the file name of the settings file (with the extension .xml) and the machine folder (a subfolder of the .VirtualBox/Machines folder), it must conform to our host operating system’s requirements for file name specifications. If the VM is later renamed, the file and folder names will change automatically.

However, if the --basefolder <path> option is used, the machine folder will be named <path>. In this case, the names of the file and the folder will not change if the virtual machine is renamed.

By default, this command only creates the XML file without automatically registering the VM with our VirtualBox installation. To register the VM instantly, use the optional -- register option, or run VBoxManage registervm separately afterwards.

VBOXMANAGE MODIFYVM

This command changes the properties of a registered virtual machine which is not running.

Most of the properties that this command makes available correspond to the VM settings that VirtualBox graphical user interface displays in each VM’s “Settings” dialog; Some of the more advanced settings, however, are only available through the VBoxManage interface.

VBOXMANAGE STARTVM

This command starts a virtual machine that is currently in the "Powered off" or "Saved" states.

The optional --type specifier determines whether the machine will be started in a window (GUI mode, which is the default) or whether the output should go through VBoxHeadless, with VRDE enabled or not;

VBOXMANAGE CONTROLVM

The controlvm subcommand allows us to change the state of a virtual machine that is currently running. The following can be specified:

- VBoxManage controlvm <vm> pause temporarily puts a virtual machine on hold, without changing its state for good. The VM window will be painted in gray to indicate that the VM is currently paused.
- Use VBoxManage controlvm <vm> resume to undo a previous pause command.
- VBoxManage controlvm <vm> reset has the same effect on a virtual machine as pressing the "Reset" button on a real computer: a cold reboot of the virtual machine, which will restart and boot the guest operating system again immediately. The state of the VM is not saved beforehand, and data may be lost.
- VBoxManage controlvm <vm> poweroff has the same effect on a virtual machine as pulling the power cable on a real computer. Again, the state of the VM is not saved beforehand, and data may be lost.
- After this, the VM's state will be "Powered off". From there, it can be started again;
- VBoxManage controlvm <vm> savestate will save the current state of the VM to disk and then stop the VM. After this, the VM's state will be "Saved". From there, it can be started again;

A few extra options are available with controlvm that do not directly affect the VM's running state:

- nic<1-N> null|nat|bridged|intnet|hostonly|generic: With this, we can set, for each of the VM's virtual network cards, what type of networking should be available. They can be not connected to the host (null), use network address translation (nat), bridged networking

(bridged) or communicate with other virtual machines using internal networking (intnet) or host-only networking (hostonly) or access to rarely used sub-modes (generic).

- Vrdeport default|<ports> changes the port or a range of ports that the VRDE server can bind to; “default” or “0” means port 3389, the standard port for RDP.

- The cpuexecutioncap <1-100>: This operation controls how much cpu time a virtual CPU can use. A value of 50 implies a single virtual CPU can use up to 50% of a single host CPU.

WEB BASED RDP VIEWER

8. WEB BASED RDP VIEWER

VirtualBox, supports Desktop Protocol (VRDP). Using VRDP we can the output of a virtual machine's window remotely on any other computer and control the virtual machine from there, as if it was running on the remote machine. VRDP is a backwards-compatible extension to Microsoft's Remote Desktop Protocol (RDP). Typically graphics updates and audio are sent from the remote machine to the client, while keyboard and mouse events are sent back. We can use any standard RDP viewer, such as the one that comes with Microsoft Windows, on Linux system, the standard open-source rdesktop program to connect to the virtual machine remotely. We should use the IP address of our host system as the server address. The VRDP server uses the standard RDP TCP port 3389 by default. The port can be changed either in the GUI VM settings or with VBoxManage modifyvm command –vrdpport option. Note that only one machine can use a given port at a time. Also on Windows hosts the default RDP port (3389) could be already used by the Windows RDP server,in this case we should choose another port for VM(s) [8].

Oracle RDPweb is an opensource RDP client that could be integrated to the webpage, Virtualbox streams the virtual machine using VRDP protocol. So, we could access those virtual machines from remote system by using some RDP Client. Here in this project we are using Oracle RDPwebControl to access the virtual machines [7].

WORKING

- We make use of Perl CGI scripts to manage the Remote Virtual Machines such as to start and stop the machine.
- The RDP client first contacts the Oracle VRDP broker (passing over any information like username, password, etc).
- The RDP broker will then contact the Oracle VRDP service on behalf of the client and will ask to startup the desired desktop.
- The Oracle VRDP service will first verify the username/password combination if client authentication is enabled on the service side
- If authentication succeeds, the corresponding desktop will be started up and the Oracle VRDP service returns the IP and optionally RDP port of the virtual machine (VM) running the desktop.

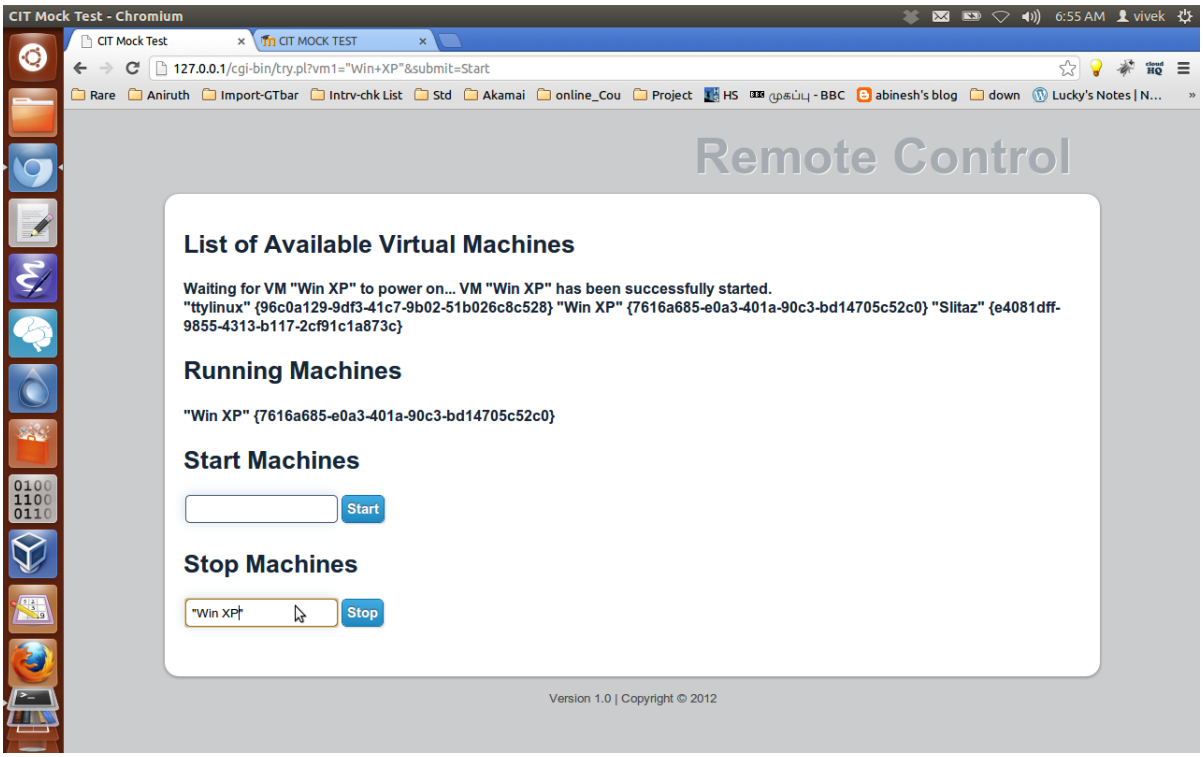

Fig.7 Control Panel to Start/Stop the Remote Virtual Machines

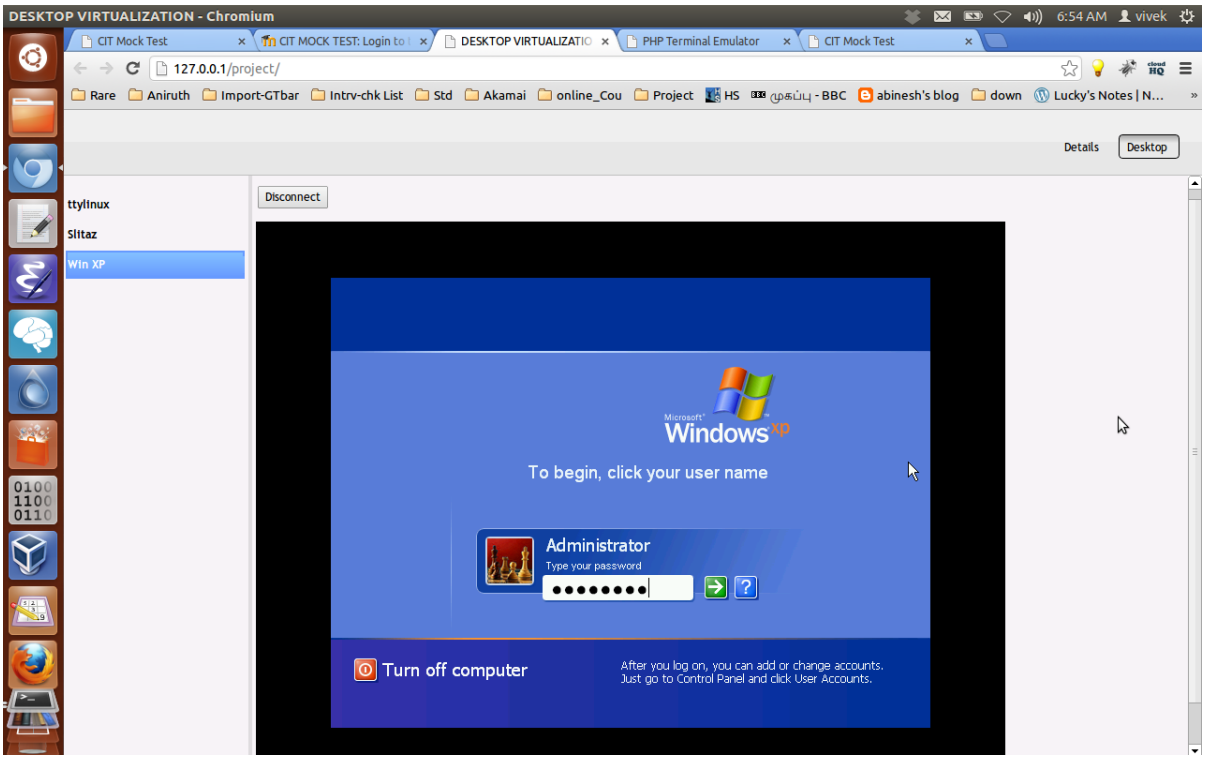

Fig.8 Windows XP Virtual Machine accessed Using RDP Viewer

- This information is used by the RDP broker to construct an RDP Server Redirection Packet containing either the VM host/IP address as the server to redirect to or a routing token containing encoded IP address and RDP port information

- The latter is necessary, because VRDP does not use the standard Windows RDP port. Thus the RDP broker needs to hand back both the IP and the RDP port information.

- Finally, this RDP redirection packet is sent back to the RDP client and the client will redirect accordingly.

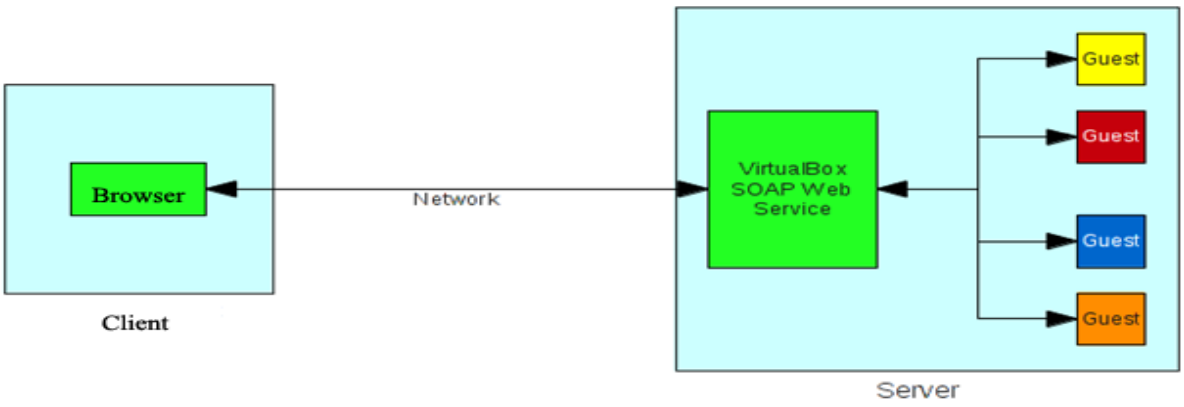

Fig.9 Working of RDP Web and Virtual box VRDP

MOODLE

9. MOODLE

Moodle is an open-source free learning management system that allows us to create powerful, flexible, and engaging online learning experiences. The phrase "online course" often connotes a sequential series of web pages, some images, maybe a few animations, and a quiz, provided online. The word Moodle was originally an acronym for **Modular Object-Oriented Dynamic Learning Environment**, which is mostly useful to programmers and education theorists. Every Learning Management System (LMS) has a paradigm, or approach, that shapes the user experience and encourages a certain kind of usage. Moodle is designed to support a style of learning called Social Constructionism. This style of learning is interactive. The social constructionist philosophy believes that people learn best when they interact with the learning material, construct new material for others [9]. Moodle allows you to add several kinds of static course material. This is course material that a student reads, but does not interact with

- Web pages
- Links to anything on the Web (including material on your Moodle site)
- A directory of files
- A label that displays any text or image

However, Moodle also allows us to add interactive course material. This is course material that a student interacts with, by answering questions, entering text, or uploading files

- Assignment (uploading files to be reviewed by the teacher)
- Choice (a single question)
- Lesson (a conditional, branching activity)
- Quiz (an online test) order for each course.

INSTALLATION STEPS

Pull the code from the Git repository

```
git clone -b MOODLE_22_STABLE git://git.moodle.org/moodle.git
```

Create database ‘moodle’

CREATE DATA DIRECTORY

Create an empty directory to hold Moodle files. It must not be in the area served by the web server and must have permissions so that the web server user can write to it. Typically, either make it owned by the web server user or give it write permissions for 'everyone'.

INSTALL MOODLE CODE

Unzip / move / copy the Moodle code (obtained above) so that it will be served by our web server (e.g. on Debian based Linux, move to /var/www/moodle)

Check the permissions and make sure that the web server does not have permissions to write to any of the files in the Moodle code directories (a very common root cause of sites being hacked). If we need to, configure our web server to serve the Moodle site with our chosen URL.

CONFIGURE MOODLE

In the Moodle code directory, find the file config-dist.php and copy it to a new file called config.php. Edit config.php with our favourite editor and change the appropriate settings to point to our site, directories and database. The Moodle install script will create config.php for us if it does not exist but make sure we set permissions appropriately.

INSTALL MOODLE

Go to the URL for our moodle site in a browser or run the command line version

```
/usr/bin/php /pathtomoodle/moodle/admin/cli/install.php
```

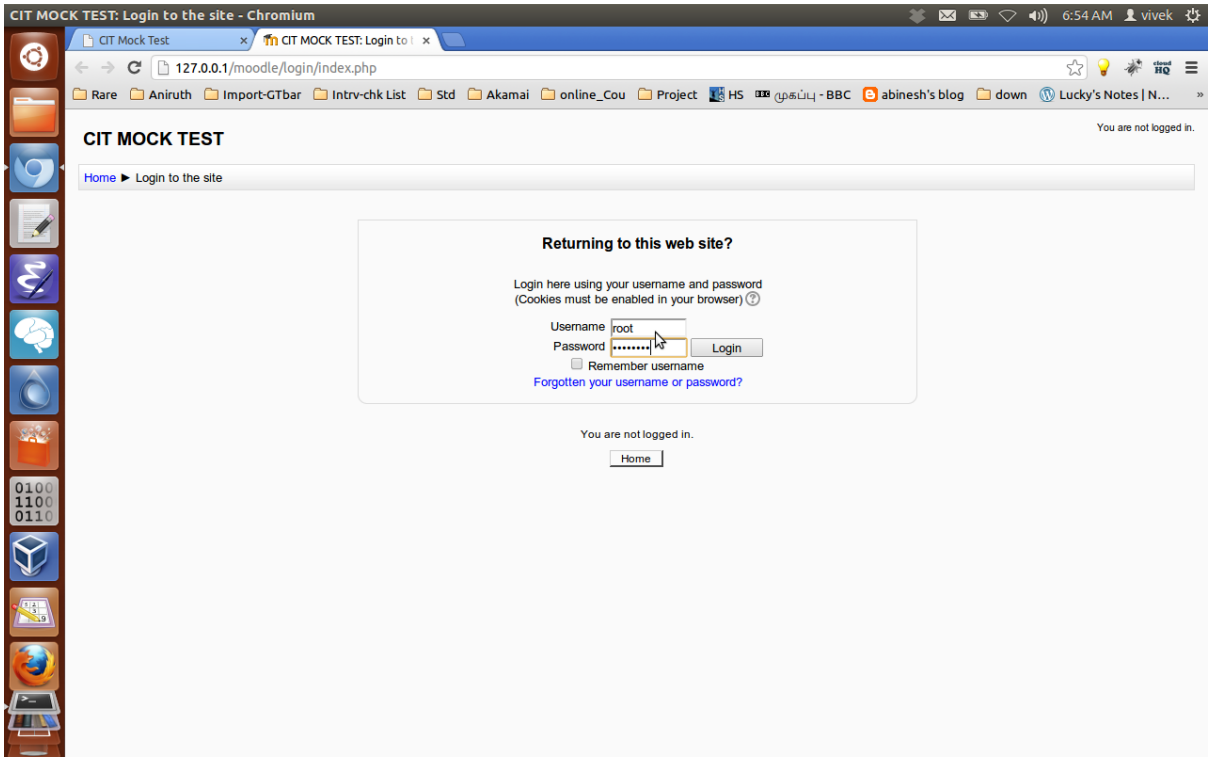

Fig.10 Moodle Login Page

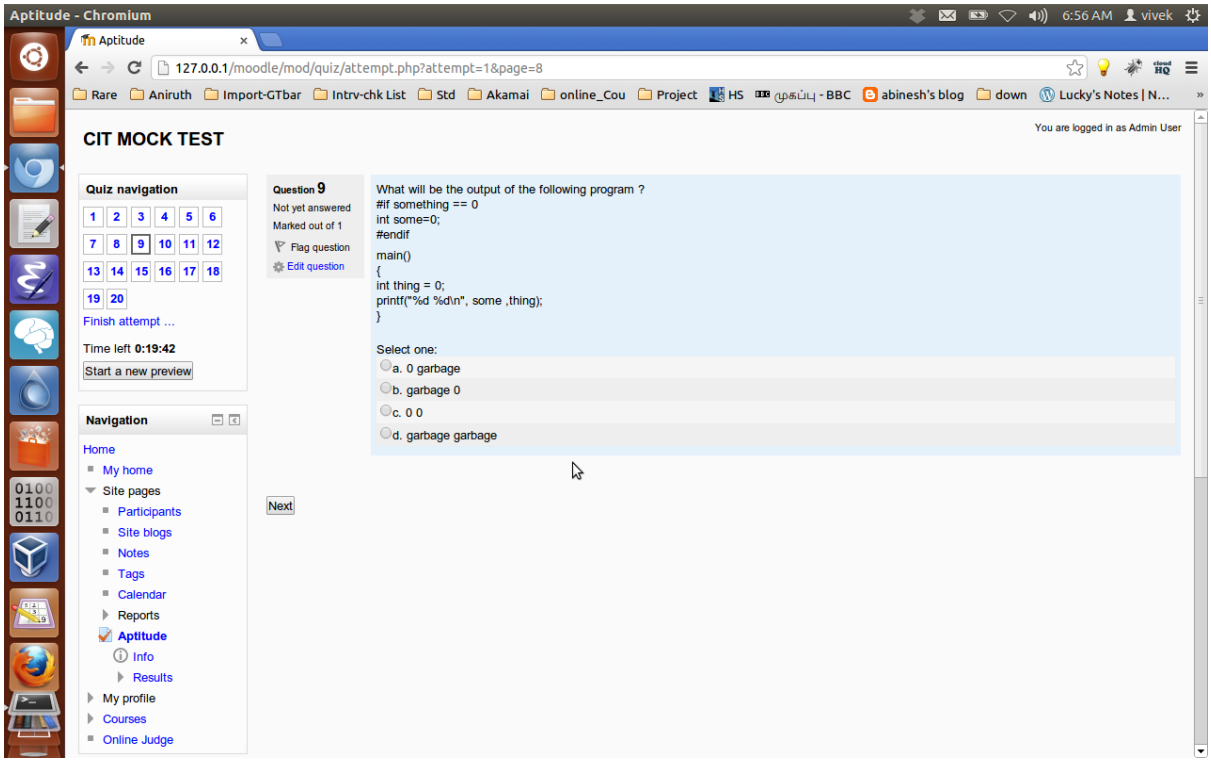

Fig.11 Moodle Quiz Page

ONLINE COMPILER

10. ONLINE COMPILER

The primary purpose behind the development of online compilers is the mobility that they provide to programmers. Because online compilers require only a web browser and connection to access and edit source code, using an online compiler has made it significantly easier for programmers to work on projects on multiple computers and/or devices. This is in contrast to conventional compilers which require programmers to set up and store their source code on a single computer. When programming for different devices or operating systems with a conventional compiler, a programmer would have to have physical access each device and/or operating system for which he or she would like to program. However with an online compiler, because the program is stored online, a programmer can work with a wide variety of devices and/or operating systems without having to physically having access to them, marking a large reduction in the hardware and software required. The GNU Compiler Collection (GCC) is a compiler system produced by the GNU Project supporting various programming languages. GCC is a key component of the GNU tool-chain. Originally named the GNU C Compiler, because it only handled the C programming language. Compiler was extended to compile C++. Front ends were later developed for Objective-C, Objective-C++, Fortran, Java, Ada, and Go among others. In this project we integrated GCC application on our web front end which supports compilation of C codes.

```
<?php
        $CC="gcc";
        $out="./a.out";
        $code=$_POST["code"];
        $input=$_POST["input"];
        $filename_code="main.c";
        $filename_in="input.txt";
        $filename_error="error.txt";
        $executable="a.out";
        $command=$CC." -lm ".$filename_code;
        $command_error=$command." 2>".$filename_error;
        $file_code=fopen($filename_code,"w+");
        fwrite($file_code,$code);
        fclose($file_code);
```

```
        $file_in=fopen($filename_in,"w+");
        fwrite($file_in,$input);
        exec("chmod 777 $executable");
        exec("chmod 777 $filename_error");
        shell_exec($command_error);
        $error=file_get_contents($filename_error);
        if(trim($error)=="")
        {
            if(trim($input)=="")
            {$output=shell_exec($out); }
        else
        {   echo "<pre>$error</pre>"; }
        exec("rm $filename_code");
        exec("rm *.o");
            exec("rm *.txt");
        exec("rm $executable");
?>
```

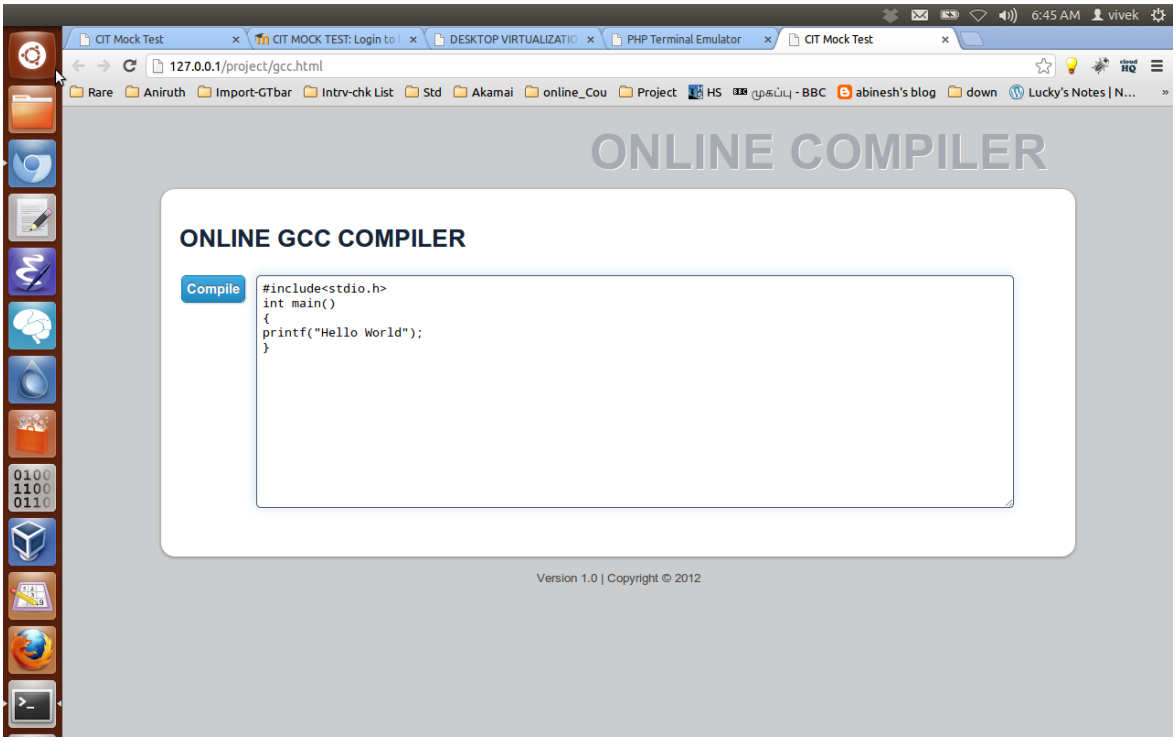

Fig.12 Integrated GCC Compiler

TERMINAL CLIENT

11. TERMINAL CLIENT

If suppose we need to make some changes in the server from remote host we need to connect to the server using ssh or telnet clients instead we thought we could integrate a web front end for the telnet client which could act as a remote terminal.

The main advantages of web-based terminal can be summarized as follows

Web-based terminal requires no local installation of client software. It is thus possible to access SSH servers through a web browser from anywhere. As communication is based on HTTP it is also possible to access SSH servers from behind a firewall or proxy that restricts Internet access to only ports 80. Web-based SSH implementations can be embedded into any web page allowing them to be integrated into other web-based applications. Many web-based SSH tools have unique features such as the ability to share terminals with other users, can display images within terminals, and other useful capabilities.

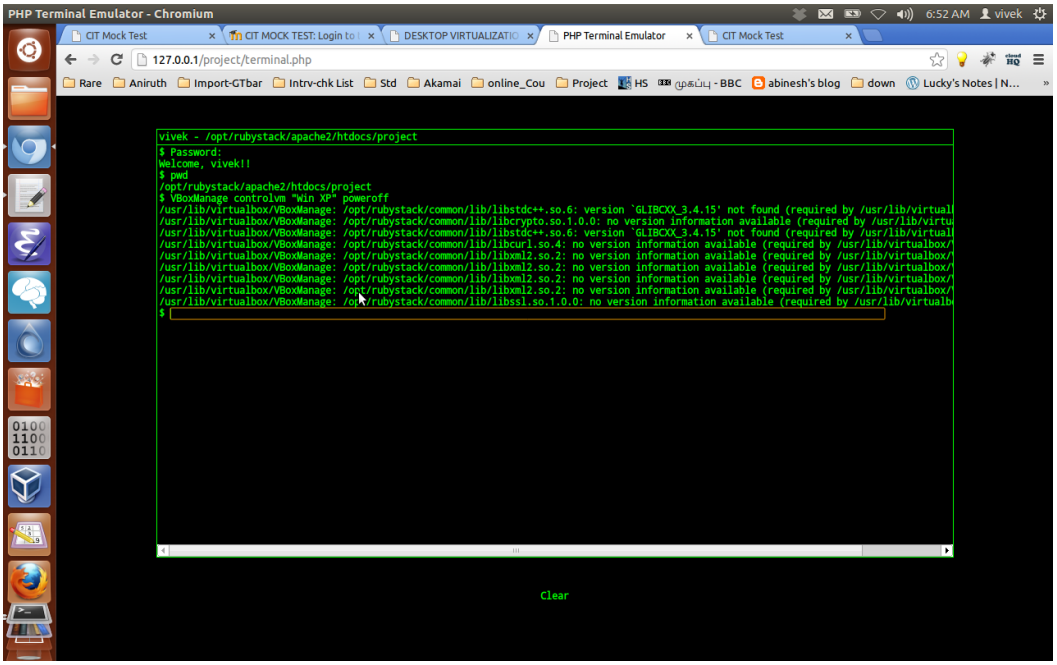

Fig.13 Integrated Terminal Client

CONCLUSION AND FUTURE WORK

12.CONCLUSION AND FUTURE WORK

In this project, we have configured a novel server based desktop virtualization system that supports clients to execute applications on their devices. This system supports multiple clients, resource sharing and centralized access to clients. Clients can run all the application located on the server irrespective of its underling architecture. Further work includes multi-server implementation and load balancing among them. More work is also needed to speed up the communication between clients and the server.

REFERENCES

13. REFERENCES

[1] *Oracle VM VirtualBox User Manual 4.2*, Oracle 2012 [Online], Available: http://download.virtualbox.org/virtualbox/UserManual.pdf

[2] *Oracle VM VirtualBox Programming Guide and Reference 4.2*, Oracle 2012, [Online], Available: http://download.virtualbox.org/virtualbox/SDKRef.pdf

[3] Dan Kusnetzky, *Desktop Virtualization and Independent Computing,* Wiley, 2008

[4] Jaime Halscott, *Desktop Virtualization and Evolving Strategies for IT Service Delivery*

[5] Hassell, J, *Server Virtualization: Getting Started*, Computerworld, May 2007

[6] *Real Benefits of a Virtual Infrastructure*, Dell Corporation, 2007, [Online] Available: http://www.dell.com/downloads/global/solutions/public/articles/Corp_Sept_enews.pdf

[7] Dale Vile, Tony Lock, Martin Atherton and Jon Collins, *Desktop Virtualization For Dummies*, Wiley, 2011

[8] *Virtualbox Perl And PHP WSDL Toolkit*, Available: http://soap-wsdl.sourceforge.net

[9] William Rice, *Moodle 2.0 E-Learning Course Development*, PACK publishing, August 2011