IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology
Vol. 3, Issue 5, May 2016

Creating A Virtual Lab

Shriya Limburkar[1],Tanaya Kavthekar[1],Hrishikesh Lavhare[1],Nikhil Khladkar[1],Shailesh P.Bendale[2]

Student, Department of Computer Engineering, NBN Sinhgad School of Engineering, Pune, India[1]

Asst. Prof. Department of Computer Engineering, NBN Sinhgad School of Engineering, Pune, India[2]

Abstract: Virtualization are one of the most trending technologies in the world of Information Technology and are gaining popularity around the world due to their cost reducing nature, flexibility and scalability. In this paper we intend to provide a software framework for cloud computing for college laboratories: system that provides a model in which operating systems are hosted by a central server. Clones of virtual machines hosting the operating systems are made available to the clients on demand. The availability of operating systems is made over a network, typically internet or intranet using the SSH protocol for communication between the client and the server. The central server will be providing a resource rich environment empowering the user to optimize resources, increase scalability by creating clients dynamically and flexibility to customize application accessibility. The Software framework, principled on server virtualization techniques that can be achieved using kernel virtual machine (KVM). The proposed framework is simple, modular and provides access to infrastructure commonly found within an academia. The software framework will also be equipped with a friendly graphical user interface and eliminates the tedious process of using command prompt for creating and managing virtual machines. The GUI enables the user to generate desired number of clients within the predefined server's client generation limits. It also permits the user to deploy variable packages to the created clients on request.

Keywords: Virtualization, Kernel Virtual Machine (KVM), Virtual Machine, Libvirt API.

I. INRODUCTION

In college laboratories often more than one operating systems are required by the students for academic assignments. For such a setup often the students multiple boot the college computers. However the process of multiple boot is lengthy and tedious process and has its limitations. Hence the paper intends to provide an alternate option of creating a virtual lab which uses virtualization technique consisting of different virtual machines hosting the operating systems the students require in the operating system.

In this paper we give a detail introduction of the technologies we have used to create a virtual laboratory such as Virtualization, Virtual Machine, Hypervisors, Libvirt API used for connecting the user to the hypervisor and SSH protocol used for client server communication.

II. VIRTUALIZATION

There's a new wind of change in the IT industry today. It's called virtualization. In a datacenter one can find virtualization at several levels but the virtualization type which is extensively adopted in the IT industry is guest operating system (OS) or server virtualization. Guest OS virtualization is a software layer that provides the ability to expose physical resources to make them available to several different virtual machines at the sametime.

Guest OS virtualization technologies come in two flavors. The first is a software layer that is used to simulate a physical machine on top of an existing operating system running on a hardware host. The second layer is a hypervisor—a software engine that exists directly on the top of hardware and eliminates the overhead of a secondary operating system.[1]

A. Hypervisor and its role

Hypervisors are defined as software tools used to form the Virtual Machines (VM's), and they create the virtualization of various hardware resources such as CPU, storage, and networking devices. Hypervisors are also called virtual machine monitor (VMM) or virtualization managers. Virtualization of cloud data centers (DCs) use hypervisors. VMware, Xen, Hyper-V, KVM etc. are various hypervisors used. Multiple OSs concurrently run on a physical system sharing its hardware using VM's. Thus, a hypervisor permits multiple OSs to share a single hardware host. Every OS seems to have the host's processor, memory, and other resources allocated exclusively to it. However, the host processor and resources are controlled by the hypervisor and in turn it allocates resources which is needed by each OS. The hypervisor ensures functional dependencies of the guest Oss.[1]

B. What is virtual machine?

Virtual Machine is a self-contained operating environment that runs on the top of hypervisor and behaves as if it is a separate computer which has a separate instance of operating system.A virtual machine is made up of several different components such as:-

1) Configuration File

A file that contains the settings information amount of RAM, number of processors, type and number of network interface cards (NICs), type and number of virtual disks for the virtual machine. This file is in the form of Extended Markup Language (XML) and is edited accordingly for all virtual machine created in the laboratory.

2) Hard disk File
Each time you create a virtual machine, the virtualization software will create a virtual hard disk, that is, a file that acts like a typical sector-based disk. In the implementation we creates an image (.img)file for each virtual machine which acts like the Hard disk file for the virtual machine.[2]

III.KERNAL VIRTUAL MACHINE (KVM)

KVM is highly-developed by Red Hat, accessible as a free, open-source alternative to other commercial solutions. VMware and virtual box products are self-generated or friendly but KVM is mainly a command-line tool. In recent Linux kernel KVM is already built in.KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware that contains virtualization extensions such as AMD-V or Intel VT.

The core virtualization infrastructure is catered by loadable kernel module and kvm.ko. kvm-intel.ko or kvm-amd.kowhich are processor specific modules also exist on KVM. Unmodified Linux or Windows images are hosted by multiple virtual machines are executed using KVM. Private virtualized hardware: graphics adapter, a network card, disk etc. is present on each virtual machine. Any improvements in new Linux kernel versions can be automatically enabled due to KVM's integration into the Linux kernel.

A Linux kernel which is new enough and has had the KVM modules built for it are used to run KVM. In case of Xen a heavily patched Linux kernel is required, on which development lags behind the mainline kernel. KVM supports QEMU Copy-on-write (QCOW) disk image format, which permits it to support a snapshot mode for its disk I/O operations. A temporary file is used as a target file for all disk writes, and original disk image file remains unchanged in a snapshot mode. The huge storage requirements associated with hosting a grid of VM's is mitigated in KVM, by running multiple VM's from one disk image. SIGKILL command is used for destroying a virtual cluster. Command is send to each hypervisor which deletes the image from disk. For Ethernet bridging, the standard Linux TUN/TAP model is supported by KVM. Each VM gets its own networking resources by using this model and making it indistinguishable from a physical machine. [3]

IV. LIBVIRT APPLICATION INTERFACE

Libvirt is an open source API, daemon and management tool for managing platform virtualization. KVM, Xen, VMware ESX, QEMU and other virtualization technologies can be managed by libvirt. In the development of a cloud-based solution these APIs are widely used in the adaptation layer of hypervisors.

Libvirt provides a suitable way to manage virtual machines and other virtualization functionality, such as storage and network interface management. Libvirt is a collection of software. An API library, a daemon (libvirtd), and a command line utility (virsh) together constitute the software.

Libvirt aims to offer a single way to manage multiple different virtualization providers/hypervisors. For instance, the command 'virsh list --all' can be used to list the existing virtual machines for any supported hypervisor (KVM, Xen, VMWare ESX, etc.) [4]

V. SECURE SHELL (SSH) PROTOCOL

Secure Shell (SSH) provides a free protocol for safeguarding network communications that isless complex and costly than hardware-based VPN solutions. Secure Shell client/server solutions deliver file transfer, command shell, and data tunneling services for TCP/IP applications. Highly protected authentication,data integrity and encryption to combat password stealing and other security threats are provided by SSH connections.[5]

VI.IMPLEMENTATION

The software framework is consists of a server and a client paradigm.

A. User Interface
The client side consists of user interface which enables the user to create virtual machines at server side and access the virtual machines using remote ssh protocol.
A simple graphical user interface consisting of radio buttons is provided for the user to choose the operating system he wish to work on.

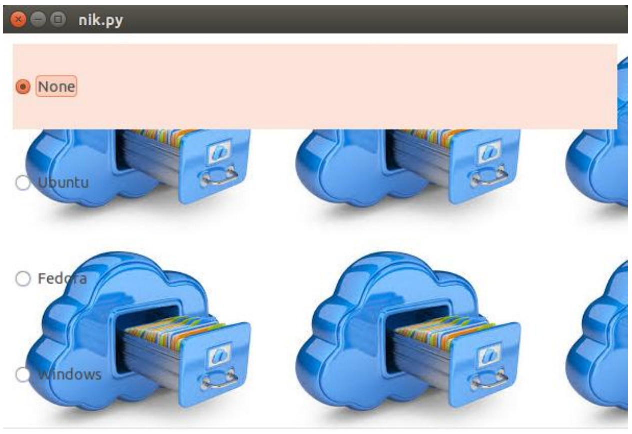
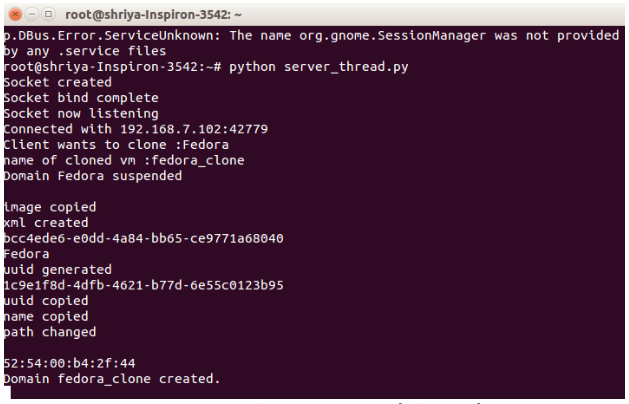


Fig 1. User Interface for selecting the required Operating System

B. Server Specification And Working
At server side the server first serves the request it gets from client. The server has a set of virtual machines hosting operating systems such as fedora and Ubuntu consisting of all the necessary software's required by the students. On receiving the input from the user the server simply clones the virtual machine which hosts the operating system the user wishes to work on.

The following snippet shows the server side working:-



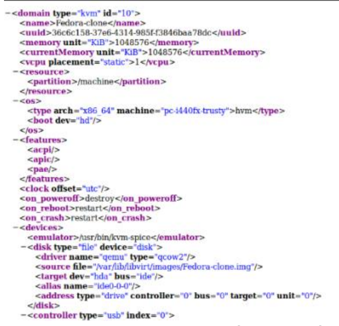
```
root@shriya-Inspiron-3542: ~
p.DBus.Error.ServiceUnknown: The name org.gnome.SessionManager was not provided
by any .service files
root@shriya-Inspiron-3542:~# python server_thread.py
Socket created
Socket bind complete
Socket now listening
Connected with 192.168.7.102:42779
Client wants to clone :Fedora
name of cloned vm :fedora_clone
Domain Fedora suspended

image copied
xml created
bcc4ede6-e0dd-4a84-bb65-ce9771a68040
Fedora
uuid generated
1c9e1f8d-4dfb-4621-b77d-6e55c0123b95
uuid copied
name copied
path changed

52:54:00:b4:2f:44
Domain fedora_clone created.
```

Fig 2. Server Side working of the software

The snippet shows how the server accepts the client request and accordingly clones the required operating system.

C. Creation Of Virtual Machine Using Xml Template

An xml file consisting of virtual machine specifications is used for creation of virtual machines. Properties such as OS type, OS version, RAM size, storage type are specified in this xml file.
The following snippet shows a template of xml file used in the implementation:



```
-<domain type="kvm" id="10">
   <name>Fedora-clone</name>
   <uuid>36c6c158-37e6-4314-985f-f3846baa78dc</uuid>
   <memory unit="KiB">1048576</memory>
   <currentMemory unit="KiB">1048576</currentMemory>
   <vcpu placement="static">1</vcpu>
 -<resource>
     <partition>/machine</partition>
   </resource>
 -<os>
     <type arch="x86_64" machine="pc-i440fx-trusty">hvm</type>
     <boot dev="hd"/>
   </os>
 -<features>
     <acpi/>
     <apic/>
     <pae/>
   </features>
   <clock offset="utc"/>
   <on_poweroff>destroy</on_poweroff>
   <on_reboot>restart</on_reboot>
   <on_crash>restart</on_crash>
 -<devices>
     <emulator>/usr/bin/kvm-spice</emulator>
   -<disk type="file" device="disk">
       <driver name="qemu" type="qcow2"/>
       <source file="/var/lib/libvirt/images/Fedora-clone.img"/>
       <target dev="hda" bus="ide"/>
       <alias name="ide0-0-0"/>
       <address type="drive" controller="0" bus="0" target="0" unit="0"/>
     </disk>
   -<controller type="usb" index="0">
```

Fig 3. Snippet of XML file of a virtual machine

VII. PERFORMANCE ANALYSIS OF SERVER

The server which hosts the virtual machine provide the virtual machines with resources such as CPU and memory. Hence it is crucial to analyze the performance of the server under increased load in terms of number of virtual machines hosted by it.
An open source benchmark platform named phoronix test suite is used for this purpose.
The Phoronix Test Suite (PTS) is anopen source benchmark platform which was used to benchmark and compare various system attributes. Of the multitude of tests and profiles available five were chosen to measure various performance attributes important in a virtual environment. These attributes include CPU usage, disk access rate, memory access rate, and how well the hypervisor is able to handle high loads distributed across several virtual machines running simultaneously.
The benchmark that we have selected for testing the server performance is Apache benchmark: measures how many requests per second a given system can sustain when carrying out 500,000 requests with 100 requests being carried out concurrently.

A. Test Scenario
For testing the performance of the server we create virtual machine with Ubuntu and Fedora as the operating systems hosted by the virtual machine.
In this scenario we go on increasing the number of virtual machines and for each step perform the Apache benchmark using phoronix.

B. Virtual Machine Specifications
Fresh images of virtual machines created solely for running these benchmarks and each virtual machine was allocated 1GB of RAM. Additionally, no other virtual machines were running on the hosts other than the machines actively running the benchmarks.
Fedora 19 and Ubuntu 12.10 LTS were the operating systems hosted by the virtual machines.
The Apache Benchmark is designed to test the performance a given server can provide. It does this by stressing a given server to determine how many requests per second it is able to handle. The version of the benchmark that is included with PTS attempts to execute 500,000 requests to the server 100 requests at a time. It then measures how many request per second the system is able to sustain.
This benchmark was chosen as it gives a good idea how the hypervisor is able to handle increasing I/O stress in terms of CPU and memory usage as the number of virtual machines
increases.

VIII. RESULTS

The following table will show the results of the Apache Benchmark test performed at the server side. The table shows the relation of number of virtual machines and average requests a server can handle.

Table I
Experimental Observations of the result of the test Apache Benchmark

Number of Virtual Machines	Average number of requests handle by the server per second.
1	12499.36
2	12373.26
3	12336.62
4	12084.15
5	11794.30

The following graph depicts the above mentioned results:

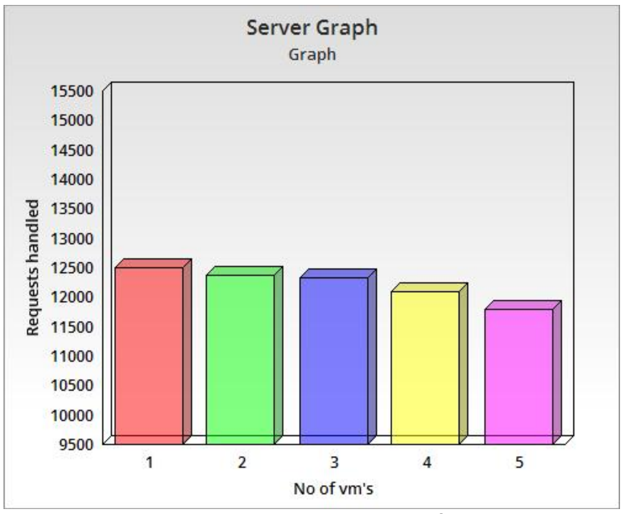


Fig 4: Graphical Representation of the result

This benchmark shows that as the number of virtual machines goes on increasing the performance of the hypervisor at the server side goes on decreasing.

IX. CONCLUSION

Thus we have discussed various virtualization techniques. Virtualization provides increased compatibility and manageability by isolating applications from underlying operating system and other applications.
The virtualization layer controls and manages the runtime execution of applications by porting applications files, configurations and settings on the target device.
We implemented a model for virtual labs which uses all the above stated perks of using virtualization techniques. In universities and colleges a large number of machines run the university stated software's and accordingly the configuration of each machine is set. However if any change in software requirements or resource requirement is stated a lot of time, resources are required to update the configurations of all the machines.
Hence using virtualization technique we have created a central server which hosts the virtual systems with required applications. And all the other machines (clients) are connected to the central servers and the applications the virtual machines are ported to the clients using virtualization techniques.

X. FUTURE SCOPE

We have implemented software using intranet connectivity. In future we can extend the accessibility of the model so as virtual machines having the required applications providedby the software can be accessed from anywhere. This can be implemented using internet connectivity.

REFERENCES

[1] Virtualisation OverviewAvailable at: http://www.vmware.com.
[2] DANIELLE RUEST NELSON RUEST "Virtualization a beginners guide, Chapter 2 Begin the Five Step Process" Pages 30-31.
[3] www3.nccu.edu.tw/~yuf/slides/kvm.pdf
[4] libivrt:http://wiki.libvirt.org/page/FAQ#What_is_libvirt.3F
[5] VanDyke "An Overview of theSecure Shell (SSH)"2008 Link:https://www.vandyke.com/solutions/ssh_overview/ssh_overview.pdf