2016 2nd International Conference on Next Generation Computing Technologies (NGCT-2016)
Dehradun, India 14-16 October 2016

CloudBox – A Virtual Machine Manager for KVM based virtual Machines

Akshay Sharma, Asif Riaz Ahmad, Divyashish Singh, Jagdish Chandra Patni
CIT, UPES Dehradun, patnijack@gmail.com

Abstract-The project aims to develop a professional Virtual Machine Manager for the KVM hypervisor. It will be a libvirt-based Web Interface for managing virtual machines. It allows creating and configuring new domains, and adjusting a domain's allocation of the underlying hardware resources. A VNC viewer will present a full graphical console to the end-users in the guest domain. To work with this service you will need the OS Linux and Web browser with installed VNC client plugin. An intuitive web interface, any system administrator can handle it with basic qualifications. The service will work with a data encrypted network connection. The Service has access only to the KVM hypervisor via the libvirt library and not to the entire OS, which improves the security of the server. End-user will have the full control of the virtualization and the virtualised hardware resource. Creating, installation, shutdown, snapshots - all this and more features are incorporated with in the virtual machines. All actions are logged. Also security features based on IAM and secure data transmission between users and the server will be incorporated in the VM manager.

Keywords: Cloud Computing, Virtual Box, VMware, Virtual Manger, Kernel based virtual Machine, Platform as a service, Software as a Service, and Infrastructure as a Service.

I. INTRODUCTION

A. Cloud Computing

Cloud computing , an IT business model, can be labelled as a new architectural paradigm for the dynamic and automated provisioning of IT (Information Technology) services assisted by the most up-to-date data centres that usually employ [11] virtualization technologies for the purpose of consolidation and environment isolation. Cloud computing delivers software (applications), platform, and infrastructure as pay per use services that are available to consumers in a utility cost model. In the industry, these services are called to as SaaS (Software as a Service), PaaS (Platform as a Service), and IaaS (Infrastructure as a Service) respectively. Most cloud computing service providers including organizations such as Google, IBM, Microsoft, and Yahoo are promptly setting out data centers in various locations around the world to deliver Cloud computing services.

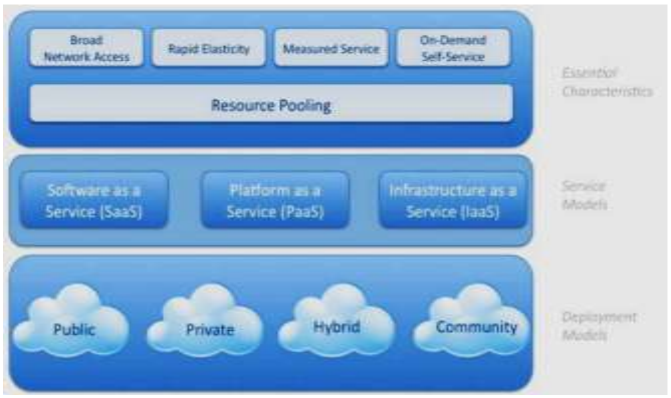


Figure 1: Cloud Computing Model

Generally the market is dominated by the shrink-wrapped software sales model. The shrink-wrapped software sales model needs that the customers are required to purchase per perusal or subscription-based license and manage the deployment by themselves, in addition of the transitioning between different versions. Hence, customers require the high initial investment and technical expertise for buying the software or the service. They also must pay an annual maintenance fee for upgrades. With the introduction of Software as a Service (SaaS), applications are moving away from ownership-based or PC based programs to web delivered hosted services.

In a model such as the shrink-wrapped software sales model, users decide the services based on their specifications without any concern as to where the services are being hosted. This model has been called as utility computing, in the beginning, or recently as Cloud computing [2]. The latter term indicates the infrastructure as a "Cloud" from which users and businesses, on demand, can use applications as services from anywhere in the world. Therefore, cloud computing can be defined as the latest paradigm for the automatic and optimal provisioning of computing services supported by latest data centers that normally use virtualization technologies for environment isolation and consolidation objectives [3]. Most cloud service providers such as Google, IBM, Microsoft, Yahoo, etc. are swiftly creating data centers in numerous places all around the globe to distribute cloud services. The model's potential can be noted from the statement by the former President of the ACM – CACM, Professor David Patterson of the University of California, Berkeley: "The Data Centre Is the Computer," [2].

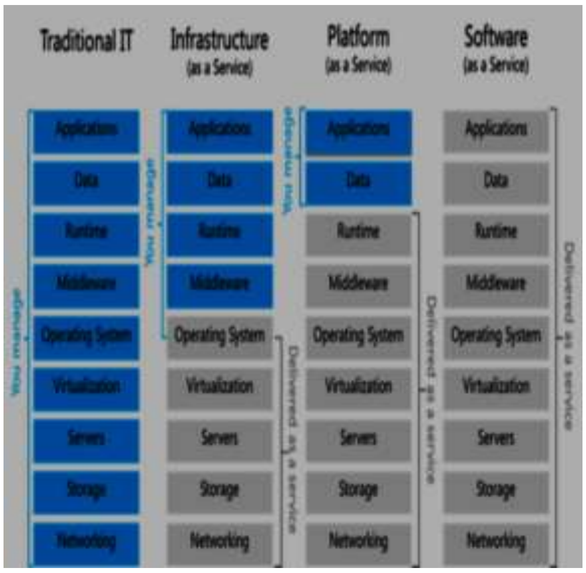


Figure 2: Cloud Computing Service Models

Developers with inventive and ingenious ideas for new web services no longer need huge capital outlays in hardware to deliver their human expense or service to fulfil it [4]. Cloud computing presents compelling profits to IT companies by releasing them from the low-level tasks such as the setting up of basic software and hardware infrastructures and thereby allowing focus to be on creating business value and innovation for their services. Several market research firms recognize the business potential of cloud computing and virtualization technology. As per Gartner, Cloud market opportunities in 2017 will be worth $350 billion. Also, many applications and services making use of utility-based computing systems such as Clouds crop up commonly as market makers or catalysts that fetch sellers and buyers together. As said by Sun co-founder Bill Joy, this generates many trillion dollars' worth of business opportunities for the utility computing industry [5]. He said that it would take some time before these markets develop to create this kind of value. Predicting now which companies will acquisition the amount is not possible. Most of them have not even started yet.

B. Kernel - based Virtual Machine (KVM)

It is a Linux kernel and FreeBSD module which enables a user-space program to access the hardware virtualization attributes of many processors. It is a full-virtualization solution for the Linux kernel based operating system on an x86 hardware involving the extensions for virtualization (AMD-V or Intel VT). It comprises of a loadable kernel module such as kvm.ko which presents a processor specific module and the core virtualization infrastructure. KVM is devised and created in the primary kernel of the Linux OS. Using KVM, anyone can execute numerous VMs running un-customized Windows or Linux images. Every VM has a private virtualized hardware: a disk, graphics adapter, network card, etc.

Kernel-based Virtual Machine is a type of hypervisor which emulates and provides for the creation of VMs on host OS. KVM presents different instruction set extensions for AMD and Intel processors and can be installed on all x86 processors.

C. KVM

In the KVM architecture, the VMs are accomplished as regular Linux processes, scheduled with the common Linux scheduler. In fact every virtual CPU shows up as common Linux processes. This allows the KVM to profit from all the attributes of the Linux kernel. Device emulation is handled by a customized variant of the QEMU that provides emulated PCI bus, USB bus, BIOS and a normal set of devices such as and network cards IDE, SCSI disk controllers, etc.

II. LITERATURE REVIEW

Virtualization Technology is a topic of deep interest in operating systems these days. It is quite useful in many scenarios: virtual test environments, server consolidation and for Linux followers who even now can't determine which version is the best [4]. Recently, hardware dealers of commodities such as x86 processors have included extensions for virtualization to the instruction set which can be employed to draft relatively simple VMMs (Virtual Machine Monitors). The KVM, or Kernel-based Virtual Machine, is a recent subsystem of Linux which uses these virtualization extensions to include virtual machine monitor (or hypervisor) ability to the Linux. Using KVM, one can run and create multiple VMs. These VMs show up as normal Linux processes and can seamlessly integrate among the rest of the system processes.

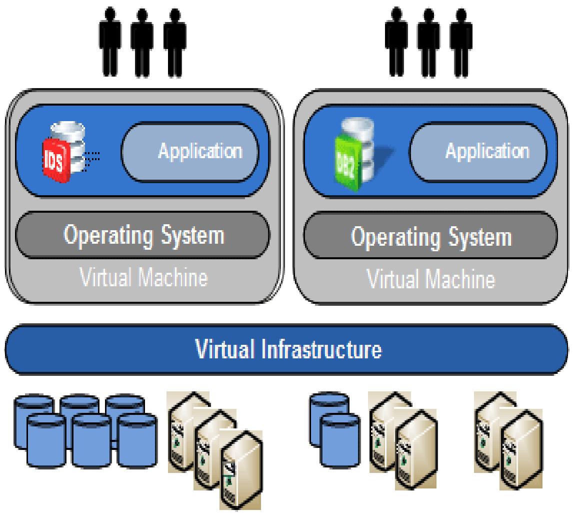


Figure 3: Virtualization Schematics

A. WebVirtMgr

WebVirtMgr[7] is a libvirt-based Web interface for managing and maintaining VMs. It enables the user to configure and create new domains and regulate a domain's allocation of resources. A VNC viewer displays a full graphical User Interface for the guest domains. Currently, KVM is the only hypervisor supported to work with this service and needs the Web browser with installed Java plug-in and Linux OS. An interactive web console/interface which any system admin can handle with rudimentary qualifications. WebVirtMgr service works along with an encrypted data connection. Service has only access to the KVM hypervisor through the libvirt library and not to the entire O. this increases the server security. Creating, shutdown, installation, full control of virtualization, snapshots – all this and more with your VMs. All activities and tasks are logged.

The features provided by this tool are :-

Host system

- ❖ Cloning Image VMs
- ❖ Manage Network Pool
- ❖ Manage Snapshots
- ❖ Manage Storage Pool
- ❖ Manage VMs Image
- ❖ View CPU Usage
- ❖ View Logs
- ❖ View Memory Usage

Virtual machine

- ❖ Create Snapshots
- ❖ Installation
- ❖ Manage ISO Images
- ❖ Shutdown/Force Shutdown
- ❖ View CPU Usage
- ❖ View Memory Usage
- ❖ VNC Access

B. Virtualization

When a generic logical definition of the underlying physical hardware is made and is cloned to make copies using a hypervisor, virtualization is achieved. Virtualization provides the required infrastructure flexibility to cloud by virtualizing the resources which allows for easy provisioning and management of these resources across hardware pools.

The various types of hardware virtualization include the following:

Full Virtualization – Almost complete simulation of the physical underlying hardware to enable the software that normally comprises of a guest OS to run unmodified.

Partial Virtualization – Only some past but not all of the targeted physical underlying hardware environment features are simulated. As a result, quite a few guest programs might require changes to execute in such a virtual environment.

Para Virtualization – A physical underlying hardware environment is not simulated; howbeit, the guest programs are run in their own separate domains, as if they are executing on separate systems. Guest programs are required to be especially customized to execute in this virtual environment.

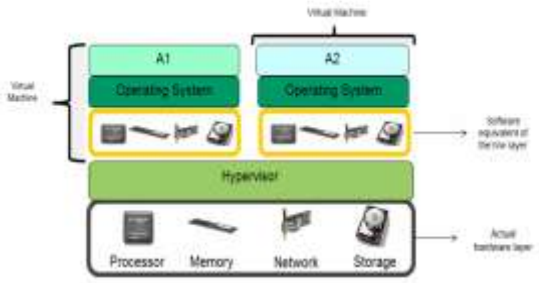


Figure 4 - Virtualization Architecture

Virtualization is generally employed for IaaS (Infrastructure as a Service). It can be easily categorized into many categories e.g. desktop, network, server, storage, etc. Quite a few types of virtualization techniques are available e.g. desktop, network, server, storage, etc. Server virtualization is the virtual partitioning of the components of a server to allow multiple virtual servers (VMs) to execute on one physical server. Employing the server virtualization for any cloud service platform is a requirement because it creates the basis and many of the needed attributes for cloud computing model.

C. Hypervisor

A hypervisor is an OS kernel that sits over the hardware and manages and maintains abstraction of the underlying hardware environment. Hypervisors are generally created to work on bare-metal or be a module in a regular operating system and in general employ the Hardware-Assisted Virtualization procedures and techniques, only if assisted by the underlying hardware systems.

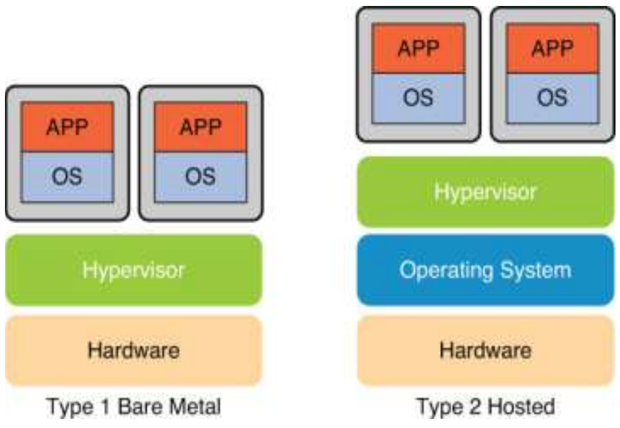


Figure 5 - Types of Hypervisors

The hypervisor is in the firmware layer and presents collaboration between the VMs executing on OS and the hardware. It is normally of 2 types – TYPE -I & TYPE –II. TYPE-I hypervisors are straight-forwardly deployed over the physical hardware and communicate with it e.g VMware vSphere, Citrix Xen Server while TYPE-2 hypervisors are deployed over the host OS in which the OS works as the medium for communication between hypervisor and underlying hardware environment e.g. Oracle Virtual Box, VMware Player.

D. Kernel-based Virtual Machine (KVM)

Kernel-based Virtual Machine (KVM) has been originally created by Qumranet, a small institution based in Israel. Red Hat then obtained Qumranet in September 2008, when KVM was ready for production. They saw KVM as the virtualization technology's next generation. At present, it is deployed as the standard VMM in RHEL (Red Hat Enterprise Linux) after version 5.4 and the Red Hat Enterprise Virtualization for Servers. Qumranet then released the source code of KVM to the open source community.

Today, well-known companies like IBM, Intel and AMD count to the list of contributors of the project. Since version 2.6.20 KVM is part of the vanilla Linux kernel and thus available on the most Linux-based operating systems with a newer kernel. Furthermore it benefits from the world class development of the open source operating system, because if Linux OS obtains improved performance via latest algos, drivers or whatsoever KVM also performs much better. KVM is a system-virtualization solution that uses full virtualization to run VMs. It has a small code base, since it was designed to leverage the facilities provided by hardware support for virtualization. KVM runs mainly on the x86 architecture, but IBM S390 and IA64 support was added only now.

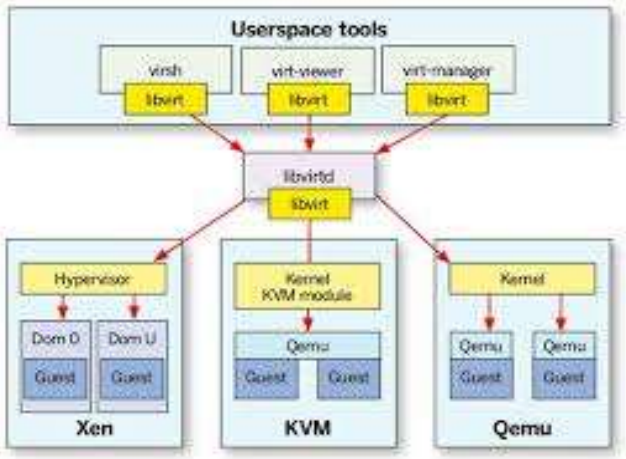


Figure 6 – KVM – Qemu Emulator

Linux has all the mechanisms a VMM needs to operate several VMs. So, the developers didn't reinvent the wheel and added only few components to support virtualization. KVM is deployed as a kernel-based bare OS module that can be loaded to extend the Linux OS by this proficiency. Thus, Linux OS is turned into a VMM. In a regular Linux OS virtual environment every process executes in one of the two modes, either user-mode or kernel-mode. KVM introduces a third mode, the guest-mode. Hence, it relies on a virtualization capable CPU with either AMD SVM or Intel VT extensions. A process in guest-mode has its own user-mode as well as kernel-mode. Thereby, it is capable of executing an OS. Such processes are representing the VMs running on a KVM host. The modes are used from a host's point of view:

Kernel-mode: Switches into the guest-mode and handles the exits due to I/O operations.

User-mode: when the guest require access to the devices i.e. I/O, etc.

Guest-mode: Executes the guest program code that is the guest OS excluding the Resource Management. In Kivity et al. [7], Kivity explained the KVM execution model and described which tasks are executed in which mode:

User-mode: The KVM kernel module is called using the ioclt() system call to execute the guest code until I/O operations admitted by the guest or an external event occurs. Such events are expressed as signals which lead to an interruption of guest code execution.

Kernel-mode: The kernel compels the hardware to run the guest code natively. If the processor exits the guest due to I/O operations or pending memory, the kernel completes the needed tasks and resumes the flow of execution. If external events such as I/O operations initiated by the guest or signals exist, it exits to the user-mode.

Guest-mode: This is done on the hardware level where the extended instruction set of a virtualization enabled CPU is used to run the native code, until an interruption is called which needs assistance by KVM, an external interrupt or a fault.

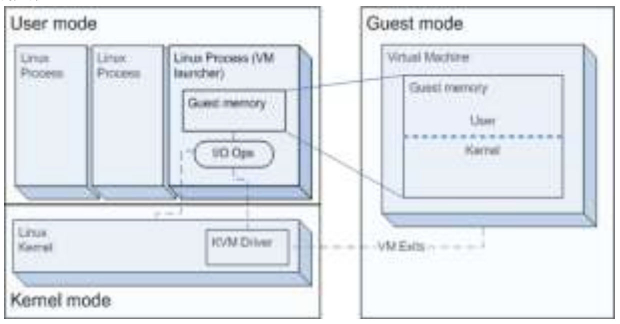


Figure 7: Modes in Linux

While a VM executes, there are a large number of switches between these modes. From kernel-mode to guest-mode switches and vice versa are very fast, because there is only native code that is executed on the underlying hardware. When I/O operations occur and the flow of execution switches to the user-mode, emulation of the virtual I/O devices comes into play. Thus, a lot of I/O exits and switches to user-mode are expected. Imagine an emulated hard disk and a guest reading certain blocks from it. Then QEMU emulates the operations by simulating the behaviour of the hard disk and the controller it is connected to. To perform the guests read operation, it reads the corresponding blocks from a large file and returns the data to the guest. Thus, user-mode emulated I/O tends to be a bottleneck which slows down the execution of a VM.

With the support for the virtio para virtual device model, KVM addresses the performance limitations by using QEMU emulated devices. Virtio is common framework to write VMM independent drivers promising bare-metal speed for these, since para virtual devices attached to a VM are not emulated any more. Instead, a backend for the para virtual drivers is used to perform I/O operations either directly or through a user-mode backend. KVM uses QEMU as such a backend which handles I/O operations directly. Thus, the overhead to mimic the behaviour of a IDE hard disk is tremendously decreased to simply using kernel drivers to performing certain operations and responding.

III. MODEL AND ARCHITECTURE

In the KVM architecture model, the VM is executed as a normal Linux process, scheduled by the normal Linux scheduler. In fact every virtual CPU shows up as a normal Linux process. This enables the KVM to profit from all the attributes of the Linux OS kernel. Device emulation is handled by a customized version of QEMU which presents emulated USB bus, BIOS, PCI bus and a regular set of devices like network cards and IDE, SCSI disk controllers, etc.

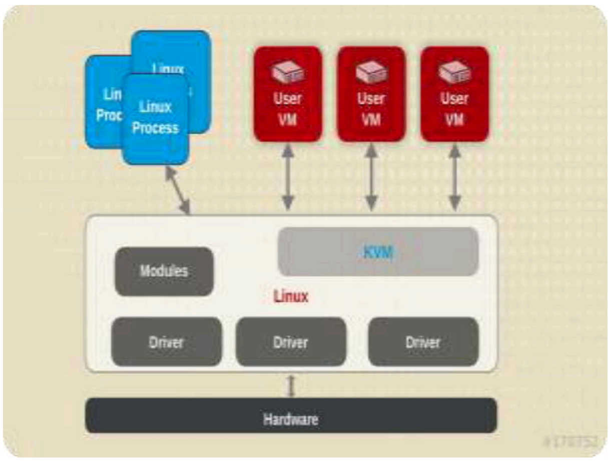


Figure 8: KVM Architecture

IV. Factors involved in KVM

A. Security

Because a VM is executed as a Linux process it avails the regular security model of the Linux OS kernel to present resource and isolation controls. The Linux OS kernel contains the Security-Enhanced Linux (SELinux), a project initiated by the US National Security Agency to increase the compulsory policy enforcement, access controls as well as multi-category and multi-level security. SELinux presents rigorous resource confinement and isolation for the various processes executing in the Linux OS kernel. The sVirt project crreates on the SELinux to present an infrastructure model to enable an admin to define regulations and policies for the VM isolation. Out of the box, sVirt establishes that a VM's resources can't be used by any other VM(or process) and this can also be expanded by the admin to determine the fine grained specifications, requirements and the permissions, for example to group VMs together to share the resource pools. The sVirt and SELinux present an infrastructure model which presents a degree of isolation and security unrivalled in the industry.

B. Memory Management

KVM acquires the dynamic memory management attributes of Linux OS. A VM's memory is stored as memory is for many other Linux OS processes and can be exchanged, assisted by the huge pages for increased performance, backed or shared by any disk/tape file. NUMA assistance enables the VMs to effectively access huge amounts of data from the memory. KVM assists the newest memory virtualization attributes from CPU vendors with assistance for AMD's Rapid Virtualization Indexing (RVI)and Intel's

Extended Page Table (EPT). Memory page sharing is assisted via a Linux OS kernel attribute known as the KSM (Kernel Same-page Merging). KSM browses the memory of every VM and where VMs have same memory pages, KSM integrates these into one page which is then shared between the all the VMs, storing only one copy. If a guest tries to adjust this shared page it will then be given its own individual copy. When centralizing too many VMs onto a single host there are many scenarios in which memory pages might be shared – for example libraries, unused memory within a Windows virtual machine, kernels , common DLLs or other objects familiar between VMs.

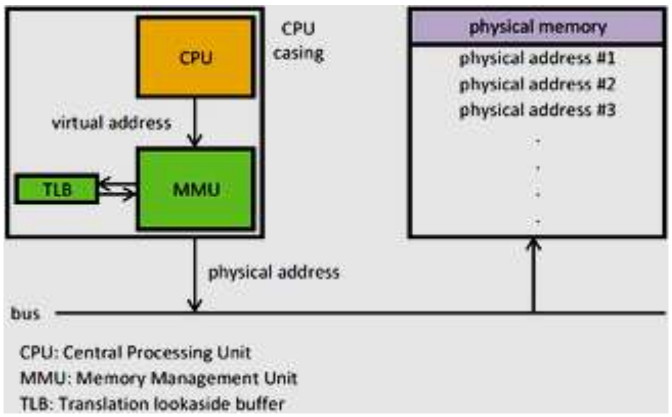


Figure 9: Memory Management Unit

C. Hardware Support

Since KVM is a component of Linux OS kernel, it avails the whole underlying hardware ecosystem, so that any hardware equipment encouraged by the Linux OS kernel could be employed by the KVM. Linux OS kernel appreciates one of the huge environments of underlying hardware dealers and the nature of the open source community, wherein the hardware providers are capable to cooperate in the improvement of the Linux OS kernel, insures that the newest hardware attributes are swiftly integrated in the Linux OS kernel, enabling the KVM to use a huge variety of underlying hardware platforms. As new attributes are joined to the Linux OS kernel where the KVM appropriates these without the increased engineering and the continued optimization and tuning of the Linux OS kernel promptly profits the KVM.

D. Storage

KVM is capable of using any storage supported by Linux OS kernel for storing VM images and ISOs, along with local disks with SCSI, SATA and IDE, Network Attached Storage (NAS) inclusive of NFS and SAN or CIFS/SAMBA with assistance for Fiber Channel and iSCSI. Multipath I/O might be used to enhance the storage throughput and to prevent redundancy. Disk ISOs and images assist thin provisioning enabling better storage consumption by appropriating storage only when it is needed by the VM rather than appropriating the whole of the storage resource at the start. The original disk format for Kernel-based Virtual Machine is QCOW2 that also has support for snapshots enabling multiple degrees of encryption, snapshots and compression.

E. Live Migration

Kernel-based Virtual Machine assists live-migration which presents the capability to transfer a powered-on VM between physical hardware hosts without any disruption to the service performance and execution. Live Migration is very transparent to the consumer/end-user. Here, the VM remains running; user applications continue to run and network connections remain active while the VM is transferred to a different physical hardware host.

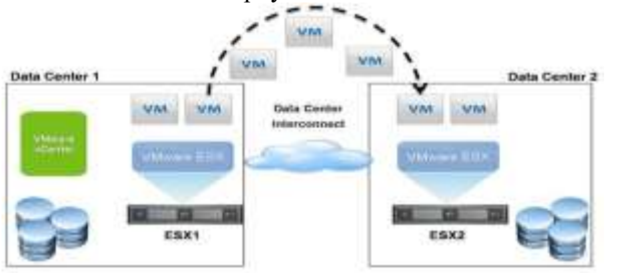


Figure 10: Live Migration

F. Guest Support

Kernel-based Virtual Machine assists a large variety of guest OS, from widespread OS such as Windows and Linux to other platforms along with FreeBSD, OpenBSD, Solaris x86, MS DOS and OpenSolaris.

G. Device Drivers

Kernel-based Virtual Machine assists hybrid virtualization whereas the para-virtualized drivers are deployed over the guest OS to enable the VMs to use an optimized and efficient I/O interface rather than the emulated gadget to convey high attainment I/O for block and network devices. The KVM hypervisor employs the VirtIO regular formed by Red Hat and IBM in association with the Linux OS community for the para-virtualized drivers that is a hypervisor exclusive interface for creating drivers for the devices enabling the identical device driver sets to be used for multiple hypervisors, enabling for improved guest portability and interoperability. Nowadays, many a hypervisor employ proprietary interfaces for para-virtualized device drivers that means that the guest images aren't interoperable amongst hypervisor platforms.

H. Performance and Scalability

Kernel-based Virtual Machine inherits the scalability and performance of Linux OS kernel; assisting the VMs with up to 256GB of RAM and 16 virtual CPUs and host systems with over 1TB of RAM and 256 cores. With around 95%-135% performance with respect to the bare metal for heavy enterprise workloads like LAMP, Microsoft Exchange, Oracle and SAP; more than one million messages/second and latency of sub-200 microsecond in VMs executing on a regular server; and the maximum concentration ratios with above 600 VMs executing heavy enterprise workloads on only one server, KVM enables even the highest demanding workloads to be easily virtualized.

I. Improved Scheduling and Resource Control

In the KVM architecture model, a VM (Windows or Linux) is a Linux OS kernel process. The VM is managed and scheduled by the regular Linux OS kernel. Over the leading several years, the Linux open source community has progressed the core of the Linux OS kernel to the point where it has industry enterprise robustness, dominant attributes, performance stability and security. The latest Linux scheduler accumulates further improvements which will enable a much finer-grain regulation of the hardware resources appropriate to any Linux OS kernel process and will enable the assurance of a QoS for any particular process. Especially, improvements including control-groups, CFS, real-time extensions and network name spaces will create the core of a kernel level infrastructure for QoS, accounting and service levels for VMs. The Linux OS kernel has a latest improved process scheduler known as the completely-fair job scheduler (CFS) to present improved process scheduling capabilities. The CFS scheduler has been expanded to count the C Groups (control groups) resource manager that allows VMs, processes, to be provided shares of the system hardware resources. Unlike other VM schedulers who give the ratios of the hardware resources to a VM established on weights, C Groups enable the minimums to be set and not just the maximums, enabling promised hardware resources to a VM but enabling the VM to employ increased hardware resources, if available.

J. Latency and Higher Determinism

Along with availing the latest resource management and process/job scheduler attributes of the Linux OS kernel to promise the network, CPU, disk IO and memory SLA for every VM, the Linux OS kernel will also have real-time appendix. These enable latency of much lower for services and applications in VMs, and a higher level of determinism that is pretty essential for mission-critical workloads for enterprise. Under this operating computing model, the Linux OS kernel processes which need a big CPU time slice are grouped into fundamental elements and processed/scheduled as per the Linux OS kernel. Therefore, the requests from VMs can be executed much swiftly, thus seriously decreasing the latency of the application processing and bettering the determinism.

The project comprises of first loading the libvirt [9] module into the OS and then using the libvirt –php API to make the backend for the project. These php modules are integrated into the front end which are designed in html/CSS/AngularJS. The libvirt-php module also helps in integrating the portal with KVM. Also the libvirt needs to be configured with the apache web browser for complete working. This involves granting root access to the Apache user group to get access to the underlying KVM modules. Also the user needs to be added to the libvirt user group. For seamless working of the portal a restart is required for the changes to take effect.

V. Problem Definition

A. Defining the problem

Virtualization provides better utilization of the physical hardware resources by abstracting the host operating system from the underlying physical hardware resources. Kernel-based Virtual Machine (KVM) enables the automatic management and self-service of the resources to provide virtualization capabilities to the consumers through various interfaces like CLI e.g. virsh, Desktop interface e.g virt manager, Web Tools e.g. WebvirtMgr, Kimchi etc. This project aims to create a user friendly web based KVM manager for easy VM management which includes provisioning/de-provisioning, scaling, editing VM's, image management.

B. Problem

To develop and implement a web based tool/portal to manage KVM based virtual machines to Provision/De-Provision, Allocation/De-allocation of resources (CPU, Memory & Storage) and manages images.

C. Assumptions

The project is designed on the assumption that the physical machine resources provided are low. The configuration of the physical machine is Intel i7-quad core CPU, 8GB RAM, and 1 TB hard disk. As we are deploying this project on a virtual machine emulated using the VMware Workstation 10. The VM is running Ubuntu 14.04 emulated with 4 virtual CPU's and 4 Gb of RAM and 20 GB of HDD. The system is capable of running at most only 2 virtual machines with decent configurations assuming the present configuration of the setup.

D. Challenges

The main challenges in deploying Web Management Portal and executing the required operations were:

- Integrating the Web portal with KVM.
- Image management with the guest OS.
- Editing virtual machine configuration.
- Integrating the noVNC client in browser for guest console access.

VI. IMPLEMENTATION

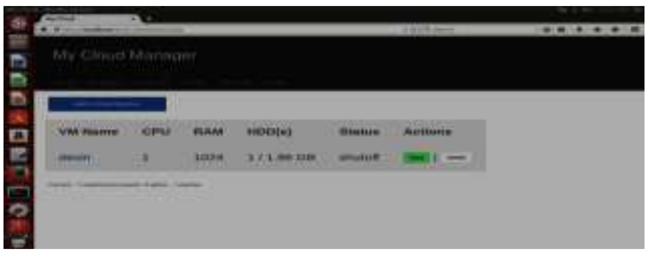
We are implementing a personal cloud manager for KVMo the Linux platform for easy and intuitive management. The VM manager will provide full VM lifecycle management, data encryption transmission for enhanced security and access management facilities. There are some pre-requisites for starting of the project. So we will start with implementing the prerequisites.

First a physical host is installed with a linux operating system. Ubuntu 14.04 in particular on VM worksatation.

Now using command line we install KVM and its supporting packages. Afterwards add your user to the libvirtd group. Also install apache MySql.

Libvirt provides many API for accessing the virtualization capabilities provided by Linux. We will be using the libvirt-php API as the base for implementing the basic functions.

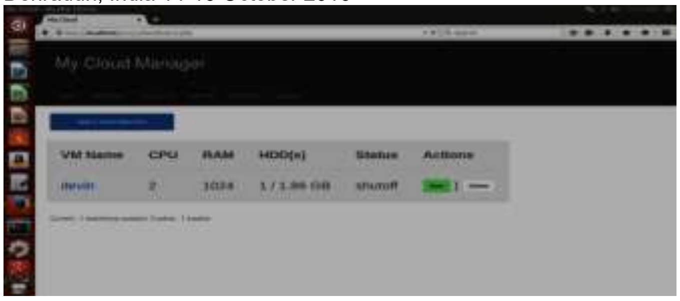
After integrating the KVM with The web portal various security modules with features for data transmission, access management, SSL will be incorporated in the Manager.



VM Name	CPU	RAM	HDD(s)	Status	Actions
[illegible]	1	1024	1 / 1.00 GB	shutoff	[illegible]

Figure 11: Dashboard

Figure 12: Dashboard after creation of a new VM



VM Name	CPU	RAM	HDD(s)	Status	Actions
devin	2	1024	1 / 1.86 GB	shutoff	[illegible]

Figure 13: Changed no of CPU for The VM devin (scaled to 2)

Figure 14: Volume v1 created

Figure 15: Volume vol1 deleted

Figure 16: Adding volume v1 to virtual machine vm1

Figure 17: Volume v1 added to the guest domain vm1

CONCLUSION

The Cloud Management Portal 'MyCloudManager" has been developed and successfully performing operations in which the following objectives are achieved:-

1. A user can connect with KVM in Ubuntu and manage guest domain.
2. The web portal provides the following Functionality
   - Provisioning/Deprovisioning of virtual machines
   - Allocation and Deallocation of resources to VM (CPU, RAM, hard disk)
   - Editing virtual machine's settings.
   - Image management.
   - Scaling the resources up/down according to demand
   - Seeing the available VM's on host and other details(active/inactive domains)

Future Scope

i) In the future the project aims to developed on a larger set of physical servers to provide more support for sustaining larger no VM's, hardware support availability, scalability etc.

ii) Also planning to develop an (android) mobile app for managing guest domains through mobile platform.

REFERENCES

[1]. L. Kleinrock. A Vision for the Internet. ST Journal of Research, 2(1):4-5, Nov. 2012.

[2]. A. Weiss. Computing in the Clouds.net Worker, 11(4):16-25, ACM Press, New York, USA, Dec. 2007.

[3]. P. Barhamet. Et al. Xen and the Art of Virtualization. In Proc. of 19th ACM Symposium on Operating Systems Principles

[4]. M. Armbrustet. Et al. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report No. UCB/EECS-2009-28, University of California at Berkley, USA, Feb. 10, 2009.

[5]. S. London. Inside Track: The high-tech rebels. Financial Times, 6 Sept. 2012.

[6]. Virt URL:http://www.ovirt.org/Documentation

[7]. WebVirtMgrUR, https://www.webvirtmgr.net/features/

[8]. Kimchi URL : https://www.github/kimchiproj

[9]. Libvirt URL: https:/libvirt.org

[10]. Libvirt-php Module URL: https:/libvirt.org/api/libvirt-php

[11]. Privacy Commissioner of Canada (OPC), "Fact Sheet: Introduction to Cloud Computing", October 2011.