Comparative Study for Selection of Open Source Hypervisors and Cloud Architectures under Variant Requirements

BELA SHRIMALI
CU Shah University,
Wadhvan City, Gujarat, India
Bela.shrimali@gmail.com

HIREN B. PATEL[2]
Sankalchand Patel College of Engineering ,
Visnagar,Gujarat, India
hbpatel1976@gmail.com

,**Abstract**

Cloud computing has been emerging as one of the prominent technologies to the end users which offers pay-as-you-go model with the help of various underlying technologies. One of such technologies is virtualization that forms the foundation for Cloud environment. Virtualization makes it possible to run multiple computing resources (in form of Virtual Machines - VMs) on the same (physical) server at same time, concurrently. Hypervisors, also known as Virtual Machine Monitors (VMM), create, manage and run virtual machines on physical machines. There are many open source hypervisors available viz. KVM, Xen, OpenVZ, VirtualBox, Lguest, LXD etc. In addition, there are many Cloud-computing software platforms available that manage the provisioning of VMs for the Cloud provider. Compatibility issues of these VMM and Cloud architectures are required to be handled for smooth functioning of the Cloud. This research aims to compare and analyze the most frequently used open source hypervisors and cloud architectures, and it would be useful to novice researchers who intend to plunge into the field of Cloud computing with an aim to use open source options for implementation of their Cloud setup.

Keywords*:* Virtual machine, Hypervisor, Virtualization, Cloud computing.

1. Introduction

A revolution in grid computing introduced a new dimension in technology known as Cloud Computing over a last decade. It satisfies the end users' need such as applications, software platforms or infrastructures over the Internet. It reduces the burden of purchasing and maintaining software or infrastructures of the end users and industries as they are available in Cloud on pay-as-you-go model. Rather than considering the Cloud computing as a new technology, it should be treated as a new operational model that brings together a set of existing technologies to run business in a different way [Zhang et al (2010)]. It uses the existing technologies like grid computing, utility computing, virtualization and autonomic computing to facilitate the end users by combining the features of all-in-one to realize resource sharing and dynamic resource provisioning [Zhang et al (2010)]. All these attributes make the paradigm popular among the researchers, developers and the end users community. The services provided by Cloud technology based on subscriptions are divided in to three categories viz. Infrastructure as a Service (IaaS) [B. Sosinskyl (2010)], Platform as a Service (PaaS) [B. Sosinskyl (2010)] and Software as a Service (SaaS) [B. Sosinskyl (2010)]. IaaS is one of the frequently and commonly used services among the Cloud users. It consist of a large number of server, disk, memory units to provide service to millions of end users. [Jing et all (2013)].

Cloud computing has a four layered architecture [Zhang et al (2010)] as shown in fig 1. Each layer has clearly identified responsibilities. First is the hardware layer, a lowest layer responsible for managing hardware resources like server, switches, power, cooling system etc. Second is the infrastructure layer, which also known as virtualization layer responsible for creating a storage and computing resources by dividing the physical resources using virtualization technologies. The infrastructure layer plays an important role since many key features, such as dynamic resource assignments are only available through virtualization technologies [Zhang et al (2010)]Virtulization of underlying hardware and software is done by creating an image/instance of physical machine called as Virtual Machine. VMM is responsible for creation of virtual machine. Third layer is platform layer which consists of operating systems and application frameworks. The upper most forth layer is application layer which consists of the actual Cloud applications [Zhang et al (2010)]All these layers are loosely coupled i.e this layered architecture handles the maintenance and management overhead of Cloud

computing. The different services like infrastructure, platform and software correspond to these layers in the architecture.

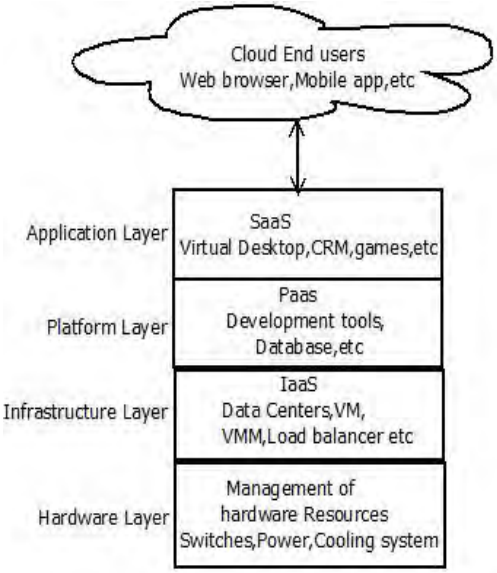

Figure 1: Layered architecture [Zhang et al (2010)]

Based on the idea of deployment models, the Cloud computing can further be categorized into four categories viz. public [Zhang et al (2010)], private [Armbrust et al(2010)], hybrid [Zhang et al (2010)] and community Cloud [B. Sosinskyl (2010)]. , When these deployment models are made available to general public in pay per use manner, it is called public Cloud [Zhang et al (2010)]Amazon Elastic Compute Cloud (EC2) [5], IBM's Blue Cloud [6], Sun Cloud, Google AppEngine [7] and Windows Azure Services [8] Platform are some of the example of public Cloud platforms. While an internal data center of organization only used by an organization not available in general can be known as private Cloud [Armbrust et al(2010)]. A hybrid Cloud is a combination of both, Public and Private Cloud. In hybrid Cloud, the organizations rely on public Cloud as needed [Zhang et al (2010)]. Cloud organized/deployed to satisfy a common purpose/aim is called as community Cloud. Figure 2 shows the deployment location of different Clouds.

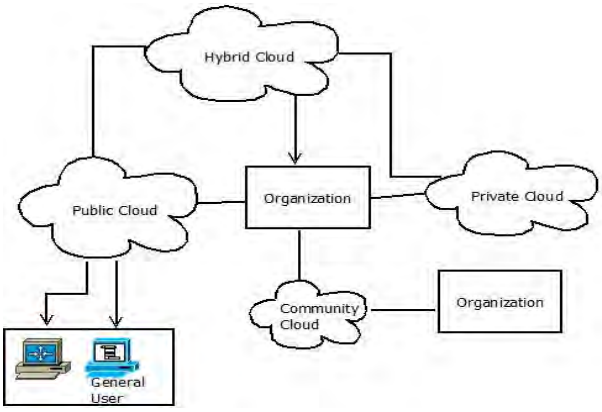

Figure 2: Deployment of different Cloud models [B. Sosinskyl (2010)].

Virtualization is considered as a most important technology from the number of underlying technologies, services and infrastructure-level configurations that make the Cloud computing possible. In a simple term, virtualization is a process of creating and managing a virtual entity. It is a mechanism which abstract the hardware and the system resources of operating system [Zhang et al (2010)] .This abstraction and virtualization are carried out across a pool of servers by Hypervisor or VMMS [Younge et al(2011)], which provide multiple isolated execution environments. It is installed on server and it provides a virtual operating platform to the guest operating systems (guest OS) and observes execution of guest OS [Sabahil (2012)]. There are several ways to implement a virtualization. Most of the virtualization strategies are fall in to four main strategies like full virtualization, para-virtualization, Operating System-level Virtualization and Native Virtualization [Walters et al (2008)].

Overviews of all these virtualization techniques are as follow:

• Full virtualization: In full virtualization, VMM will give the image of physical machine to the each VM request. In this way it executes the unmodified OS to execute the privileged instructions. It is successful as it provides isolation of users from each other and from the control program [Mishra and Jaiswali (2012)]. Some examples of full virtualization are QEMU, Microsoft Hyper-V, Oracle VirtualBox, VMware Workstation, VMware Server, Microsoft Virtual PC.

• Para virtualization: In para-virtualization, each VM has an abstraction of the hardware that is similar but distinguishable to the underlying physical hardware. Guest operating systems are modified to execute VMs. As a result, the guest operating systems are executing on a VM to provide a near-native performance. Para-virtualization methods are still being developed and thus, have limitations including several securities issues like the guest OS cache data, unauthenticated connections and "guest" OS needs to be modified [Abels et al (2005)]. Disaster recovery, migration, capacity management are the benefits of para-virtualization [Mishra and Jaiswali (2012)]. Some examples of para-virtualization are Xen, KVM, UML VM.

• OS-level Virtualization: Unlike both para-virtualization and full virtualization, OS-level virtualization does not depend on a hypervisor. Instead, it modifies the operating system securely to isolate multiple instances of an operating system within a host machine. The advantage to OS-level virtualization is performance of near-native speeds. No hypervisor/instruction is required. Some examples of OS-level virtualization are OpenVZ and LXC.

• Native Virtualization: In native virtualization, multiple unmodified operating systems are allowed to run alongside one another. In this technique operating systems are capable of running on the host processor directly. That is, in native virtualization, it does not emulate a processor. Unlike the full virtualization technique where it is possible to run an operating system on a fictional processor, it gives full part of processor to unmodified OS, which leads to a poor performance. Some examples of native virtualization are Intel and AMD supports virtualization through the Intel-VT and AMD-V virtualization extensions [Walters et al (2008)].

However, two leading ways two implement server virtualization by VMM/hypervisor are full virtualization [Walters et al (2008)] and para-virtualization [Abels et al (2005)]. Also, there are three VMM/hypervisor models as shown in fig 3.

• Type 2 VMM.

• Hybrid VMM

• Type 1 VMM

The type 2 VMM runs within a host operating system. All guest environments are operated above the level of host OS. Examples of these guest environments include the Java Virtual Machine and Microsoft's Common Language Runtime. The hybrid model is used to implement Virtual PC, Virtual Server and VM Ware GSX. These depend on a host operating system that shares control of the hardware with the VMM. The type 1 VMM controls the guest operating systems instance execute at an upper level of it with hardware. Xen and Windows Server virtualization are the examples of type 1 hypervisor.

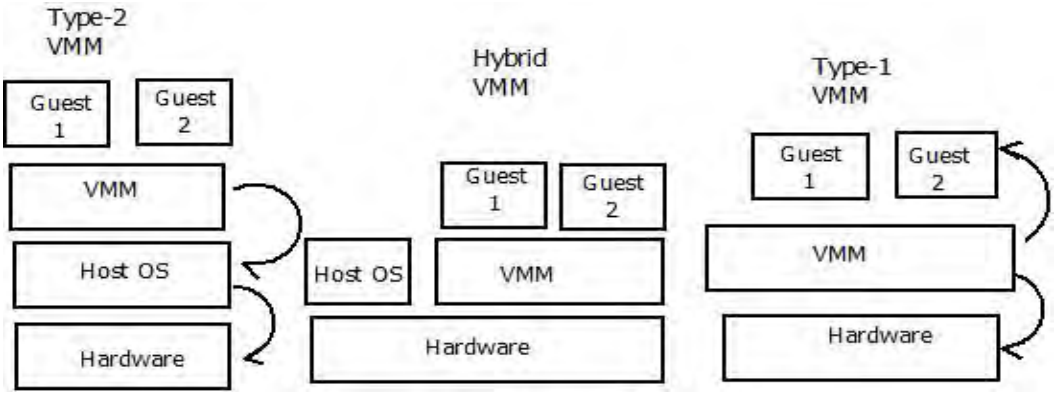

Figure 3: hypervisor [Walters et ali (2008)].

The vital component of any virtualization technique is the choice of hypervisor [Fenn et al (2009)]. The different VMMs provide a framework which allows VMs to run. Each of these Cloud frameworks with VMM relies on a library called libvirt, which is designed to control the start and stop of VMs. However, this abstraction is not fixed, as there is a unique option for each VMM that needs to be set. For every different VMM and its version, there is a different input to libvirt library. For this and other reasons, the different Cloud frameworks support different subsets of the hypervisors [Sempolinski and Thain(2010)]. VM performance is a crucial component of overall Cloud performance. Once a hypervisor has added overhead a higher layer cannot

remove it, so it reflects to the overall Cloud performance. An alternative to virtualization in the Cloud is provided by Container-based virtualization [Soltesz et all (2007)]. As Cloud can be considered as a pool of servers where the end user asks for the services with desired VM configuration and it is available on pay per use basis. If the Cloud paid services are used for the organization with the large number of users, it will be more cost effective then implementing a private Cloud. Open source frameworks are contemplate while selecting the open source which must be flexible enough to work with many underlying systems [Sempolinski and Thain(2010)], whereas commercial Clouds only need their system to work with the hardware that they have. Therefore, open source provides freedom to choose the best option. The role of Cloud hypervisors/VMM and Cloud architecture in Cloud environment can be viewed from the figure 4.

In the figure Cloud environment is shown with open source hypervisors and Cloud architectures. Hypervisor creates VM in Cloud while Cloud architecture will allocates VM in Cloud. Here, the study is carried out with the motto of answering the following questions for selecting best option:

• Suitability of different VMM in Cloud computing.

• Compatibility of Cloud platforms with VMM.

Therefore, in the present paper, firstly we briefly address the basic architecture of open source VMM with different container based virtualization technologies and different open source Cloud platforms following comparisons among them.

The rest of the paper is organized as follows. Section 2 describes various Hypervisor/VMM with comparative study followed by different open source Cloud architectures which enlightens the compatibility of hypervisors with Cloud architecture in Section 3, followed by conclusion and future work in section 4 and the list of references used in the paper is provided in section 5.

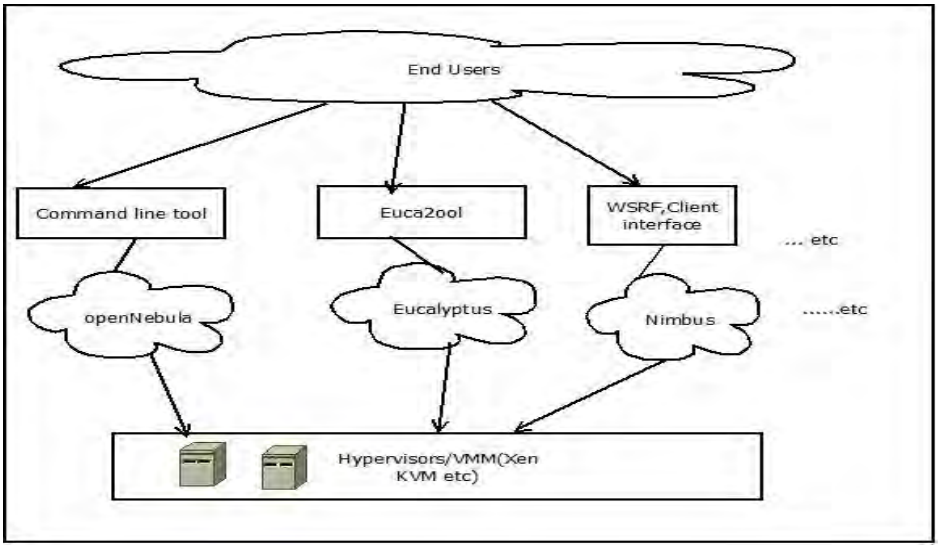

Figure 4: Cloud environment

2. HYPERVISORS

In the previous literature reviews [17 , 18 , 19 , 20 , 21 , 19 , 22 , 13], some of the open source hypervisors (like Xen, KVM etc.) are discussed either separately or they are compared (on some characteristics like performance, scalability etc.) This section includes discussion on few open source hypervisors available viz. KVM, Xen, OpenVZ, VirtualBox, Lguest, LXD, with the container based virtualization technology like OpenVZ and LXC and then we compare all these virtualization technologies by introducing more attributes.

Following section discussion covered on open source VMM called KVM, Xen, OpenVZ, VirtualBox, Lguest, LXD, with the Container based virtualization technology like OpenVZ and LXC.

1. KVM [Kivity et al (2007)]: KVM is fully integrated virtualized, simple and easy to use, solution for Linux. Its integration to Linux allows the usage of the large set of Linux features set with maximum advantage. It is attached in the mainstream Linux kernel since version 2.6.20. Device node (/dev/kvm) is open to create Virtual Machine in KVM [Franssens et al (2013)].The default scheduler for KVM is the Linux priority-based scheduler [Gu and Zhao (2012)]. As KVM is a kernel module, Linux scheduler schedules VMs as regular processes. Also, KVM can be introduced as an extension of generic and open source machine emulator QEMU with support for the x86 VT extensions. In which virtual machines can make system calls without unnecessarily invoking the host kernel. Thus, potentially saving two contexts switches [Van Doorn(2006)].

KVM requires a recent Linux kernel with the KVM modules enabled. Virtualization system used by KVM differs from other virtualization technologies that require heavily modified kernels, whose development is not often current with the mainline kernel [Fenn et al (2009)]. It is a type 1 of VMM [26].

KVM [Kivity et al (2007)] Architecture: As shown in KVM architecture in figure 5, processes create virtual machines. The integration of key virtualization technology at the processor level by both Intel (Intel VT) and AMD (AMD-V) have enabled virtualization to be deeply integrated at the Linux kernel level which creates many significant benefits of KVM in terms of performance, scalability, and security.

It also incorporates Linux security features including SELinux(Security-Enhanced Linux) to add access controls, multi-level and multi-category security as well as policy enforcement[Kivity et al (2007)].

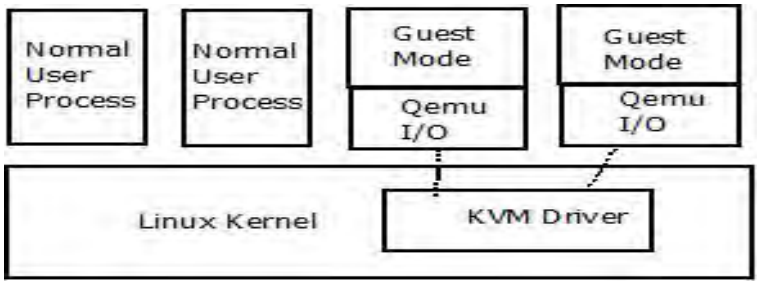

Figure 5: KVM Architecture [Kivity et al (2007)]

2. Xen[Barham et al(2003)] : Xen[Barham et al(2003)] is an open source type 1 VMM which uses full virtualization and para virtualization techniques for virtualization[Abels et al(2005)]. It is being integrated by a number of users like RedHat, Suse, XenSource and Virtual Iron [Abels et al(2005)]. Xen Architecture: Xen Architecture is shown in figure6.

As it is shown, the architecture of Xen 3.0 hosting two VMs called Domain 0, VM 1. This architecture includes the Xen VMM, which virtualize the underlying physical hardware to provides hardware access for the different virtual machines. As Xen hypervisor doesn't aware of underlying hardware of system it uses domain 0 to manage hardware access. Domain 0 is the only instance of operating system in Xen environment and thus it has direct access of underlying hardware of a machine. As, domain 0 has a special privileges, it can only access the control interface of the VMM through which other VMs can be created, destroyed, and managed. Software for Management and control runs in Domain 0. Administrators can create virtual machines with different defined privileges such as direct access of hardware. Xen has a small memory as it uses a micro kernel design and restricted interface to guest, which make it more secure and robust compare to other hypervisors.

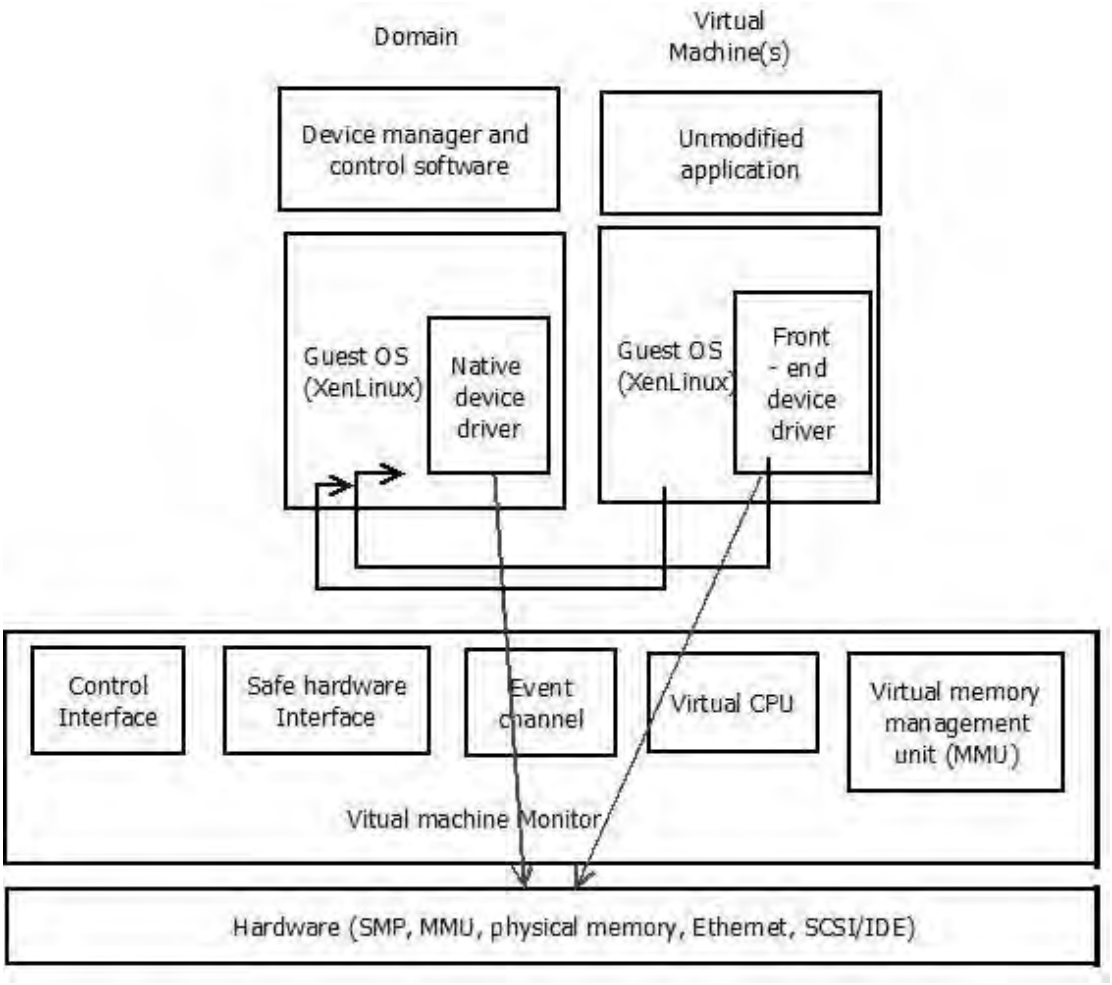

Figure 6: Xen Architecture [Barham et al (2003)]

3. Lguest [28]: Lguest is simplified Linux-on-Linux para-virtualization hypervisor that allows the execution of multiple Linux kernels on the top of a system that runs Linux kernel. It develops to serves as an educational hypervisor which supports only x86 environments. It allows Linux kernel to run internally in Lguest virtualization environment as a process runs in user space. High level architectural overview is shown in fig 7. As shown in figure, architecture can be defined with two spaces: host kernel space and host user space. It has five major components: Launcher, /dev/lguest, guest kernel, host kernel, switcher. The role of each of this components are as follow:

• Launcher: It is a host user space program which sets up, runs and services the guest.

• dev/lguest: Launcher uses a dev/lguest as a read and write character device to perform read, write operations.

• Guest kernel: It handles the guest kernel implementation.

• Host kernel: It handles the different guest kernels.

• Switcher: Switcher is the program code required to run when the switch over required between main memory and virtual memory. Lguest has benefits of having simple, easy to understand and stable architecture which makes it popular for educational purpose.

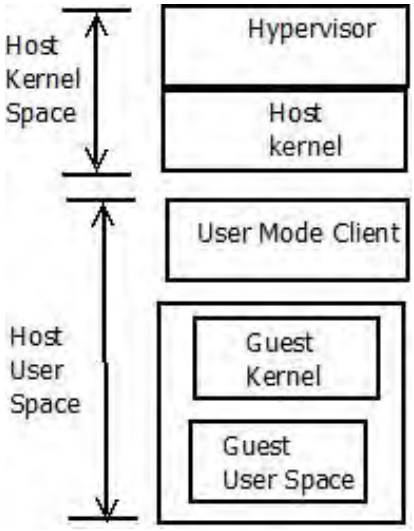

Figure 7: Lguest[28]

4. VirtualBox [Fuertes and Vergara(2007)]: VirtualBox was originally developed by Innotek, which later on acquired by Sun Microsystems in 2008. It is X86 virtualization tools that implements full virtualization. VirtualBox runs on a large number of 32-bit and 64-bit host operating systems. It is a so-called "hosted" hypervisor which allows running virtual machines created on one host and executing on another host with a different operating system. It allows executing an guest OS without modification [P. Li(2010)]. VirtualBox provides a groups feature that enables the user to organize and control virtual machines individually and in group also. It has different builds for different host operating systems like Windows, Mac OS X, Linux and Solaris hosts [Kovari and Dukani(2012)]. Thus, VirtualBox can be called as general purpose full virtualizer for server, desktop and embedded use.

2.1 Container Based Virtualization Technologies: Following are the container based technologies that uses a OS level virtualization. The Explanation of these technologies are as follows:

1. OpenVZ[Bazargan et al(2012)]: Based upon "container" technology, OpenVZ is not a true virtualization application.

It uses OS-level virtualization technology in which multiple isolated execution environments available within a single operating system kernel. This technology does not allow to run different kernels from different Oss at the same time. OpenVZ is container-based virtualization for Linux. So, it has a modified Linux kernel. Also, it permits a physical server to execute and control multiple isolated operating system instances, known as containers, Virtual Private Servers (VPSs), or Virtual Environments (VEs). As it creates multiple secure and isolated containers on single physical machine, each container executes like a stand-alone server. OpenVz has a limitation where it can requires both the host and guest OS to be Linux. OpenVZ manage the resources by handling user bean counters, I/O schedulers, disk schedulers and CPU schedulers. It does not require a reboot when these resources needed to be change during run time that makes it an attractive feature for developers and testers. OpenNode virtualization Platform can be used with OpenVZ. As shown in figure 8, OpenVZ creates many isolated, independent instance of operating system known as virtual environments or containers. It is faster and efficient as it does not have any hypervisors overhead [33].

2. LXC [34]: Like OpenVZ, LXC is a container technology, existing as a user space interface for the containment features in the Linux kernel. It does not provide a virtual machine, but rather provides a virtual environment that has its own CPU, memory, block I/O, network, etc. However, As LXC takes root in the host kernel; there is no need for a separate kernel. Like OpenVZ, LXC uses the resource management and check pointing of the host kernel. It composed of a variety of container templates, different containers managing tools, different bindings for the liblxc library (libvirt is considered an alternative library), and multiple languages (Ruby, Python, Go, Lua, etc). It achieves better performance than OpenVZ[Soltesz et all (2007)].

It has less I/O overhead and more storage capacity as it has operating system level virtualization [35]. As shown in figure 9 only one kernel execute on the host, not a different kernels for each virtual machines as Xen/KVM.It uses a chroot and linux control group at kernel level which provides more isolation then chroot and provider container isolation and controls resource and process limits.

3. LXD [Zaidenberg(2012)]: LXD is the next-generation of container hypervisor for Linux from Canonical. It aims to give rapid provisioning of resources and instant guest boot. It supports a full virtualization technology. It is compatible with OpenStack and is a most suitable pair for Linux-oriented private Cloud. It

does not support live migration. Architecture of LXD is same as LXC like it has containers, container snapshots and images. Refer figure9 for same.

2.2 Discussion On VMM: Firstly, we will discuss the widely used open source hypervisors- Xen and KVM and then focus on other too. During analyzing literature available [Younge et al(2011)], it has been identified that experiments performed for HPC (High Performance Computing) shows that Xen is suitable and most widely used hypervisor, especially within academic Clouds and grids where, KVM is identified as optimal choice for general deployment in an HPC environment. The most striking difference between the two systems is in scalability. KVM had significant problems with guests crashing, as the number of guest increased [Deshane et al (2008)]; While, KVM do not face that problem of scalability. KVM had significant problems with guests crashing in the scenario where the multiple guest trying to access many virtual hard disk frequently at the same time because KVM has high hard disk overhead and poor linearity [Rahma et al(2013)]. But, in the other case of network and CPU virtualization KVM do not face that problem of scalability. While, from the experiments carried out in [Deshane et al (2008)], Xen found more scalable compare to KVM as it is able to share resources among many guests. Performance wise, Xen was found lacking in Compared to KVM. It does not found as the best choice for enduring quality of service on infrastructure. However, as Xen is using para-virtualization technique, it does not depend on any type of CPU instructions for virtualization support. As all supervisors calls are replaced by hyper calls in to Xen hypervisor and thus, the guest operating system (OS) is modified. Still in comparison with Xen it has been found that KVM is a more promising hypervisor for grid computing. It is simple to maintain and easily deployed (compared to Xen) and also give excellent performance for many tedious jobs. Comparing Lguest with other hypervisors it has been found that it is slower than other hypervisors, though sometimes only it is noticeably, so we can say that it depends on workload. In case of VirtualBox, it has a limitation that it allows only 16GB maximum memory allotment for individual guest VMs, which actually limit the use of VirtualBox in HPC environment. Also, OpenVZ and LXC cannot use in heterogeneous environment as it require Linux as aboth guest and host OS. Summary of a discussion on VMM is shown in table1.

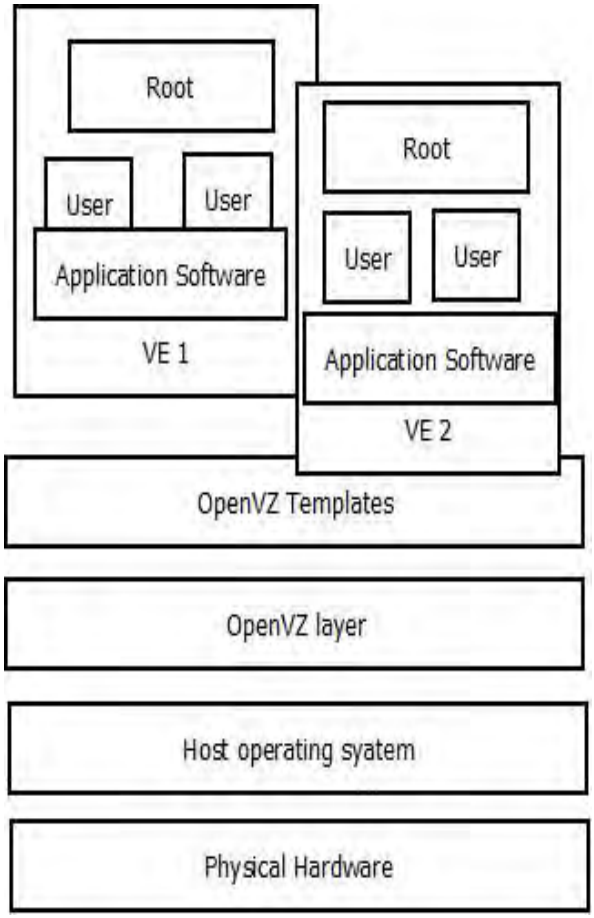

Figure 8: OpenVz architecture [33]

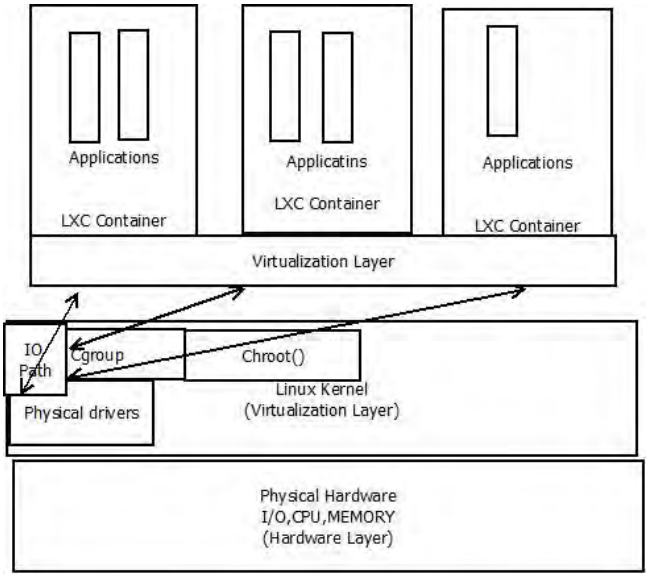

Figure 9: LXC architecture [35]

Table 1: Comparison of Cloud architecture

Hypervisor	Xen	KVM	LXD	LXC	Lguest	VirtualBo	OpenVZ
Type of VMM	Type- 1	Type-1	Container hy- pervisor	Container technolog y	Type-1	"hosted" hy- pervisor	Container technology
Type of Server Virtualizatio n	Full virtual- Ization and Para virtual	Para virtual- ization	Full virtual- ization	Operating system level virtualiz a- tion	Para virtual- ization	Full virtual- ization	OS- level vir- tualization
Required Host	Linux,BSD ,and unix	NetBSD, Linux,	Linux	Linux	Linux	Windows, Linux	Linux
Guest OS Sup- port	XBSD, Linux, Solaris,	Linux, Win- dows	Linux	Linux	Linux	Linux a nd	Linux
Suitable with	Yes	Yes	No	Yes	No	Yes	No
Service Provider/Clo ud platform	Eucalyptus, OpenNebu la, Nimbus, Amazon EC2 for on de-	Open Nebula, Amazon EC2 for on de- mand service, Enomaly	OpenStack	No hypervi- sor	Xen, KVM	OpenNebul a, Enomaly	No hypervi- sor

3. OPEN SOURCE CLOUD ARCHITECTURES

Open source Cloud computing option is one of the most important features for the Cloud to be seen as dramatic rise in the academic and business world. We have referred and tried to analyze and compare various open source options [15 , 36 , 37 , 38 , 39 , 40 , 41 , 42] of Cloud.

Various Architectures: In this subsection, we discuss Eucalyptus[43], OpenNebula [Sempolinski and Thain(2010)], nimbus [Sefraoui et al(2012)], ,OpenStack [Endo et al(2010)], Enomaly [Rahma et al(2013)], Apache VCL [Rahma et al(2013)], TPlatform [Rahma et al(2013)], XCP [Rahma et al(2013)], cloud

stack[Nurmi et al(2009)] and Ganeti[Nagar and Suman(2014)]. are also discussed in detail with its architecture.

1. Eucalyptus [Peng et al (2009)]: Eucalyptus Systems is an open-source software infrastructure for building a Cloud Computing on clusters derived from the first letter of "Elastic Utility Computing Architecture for Linking Your Programs to Useful Systems". It is used to build private, public or hybrid Clouds. It provides an integrated set of application programming interfaces (APIs) that are compatible with Amazon Web Services, including Amazon's Elastic Compute Cloud (EC2), Amazon Simple Storage Service (S3), and Amazon Elastic Block Store (EBS) [Kranasi et al (2012)].

Architecture: The five components of Eucalyptus are grouped into three separate levels. User can access the system via web interface or its own tool euca2ools for frond end interaction [Sempolinski and Thainl (2010)]. Architecture of Eucalyptus is shown with its components in figure10. The brief explanations of these levels are as follow:

• Cloud Level: It has two components: Cloud Controller (CLC) and Scalable Object Storage (SOS).
CLC provides the web user interface (an http server on port 8445) and implements on Amazon EC2 APIs. The CLC accepts user API requests from command-line interfaces like euca2ools or GUI-based tools like Management Console and manages the underlying computing, storage, and network resources. Only one CLC can exist per Cloud. It handles high-level activities like authentication, accounting, reporting and quota management. Scalable Object Storage (SOS) is equivalent to AWS Simple Storage Service (S3). It provides a basic storage implementation, known as Walrus, which is used to implement S3 APIs and is useful to store VM Images and user storage using S3 bucket put/get abstraction. EC2 provides a virtual computing environment that enables a user to run Linux-based applications [Keahey (2009)].

• Cluster Level (i.e, Availability Zone): There are two components: Cluster Controller (CC) and Storage Controller (SC). CC is used to manage the collection of resource nodes. The Cluster Controller (CC) acts as the front end for a cluster within Eucalyptus Cloud and communicates with the Storage Controller (SC) and Node Controller (NC). The CC manages instance (i.e. virtual machines) execution and Service Level Agreements (SLAs) per cluster. DCM is provided by Ubuntu Eucalyptus CC package. The SC communicates with the Cluster Controller (CC) and Node Controller (NC) within the distributed Cloud architecture and manages block volumes and snapshots to the instances within its specific cluster. If an instance requires writing persistent data to memory outside the cluster, and these data requires to write to the back end storage, which is available to any instance in any cluster [Sempolinski and Thain(2010)].

• Node Level: Node Controller (NC).It hosts the virtual machine instances and manages the virtual network endpoints. The NC downloads and caches images from Scalable Object Storage as well as creates and caches instances [Sempolinski and Thain(2010)].

2. OpenNebula: OpenNebula is a flexible tool that coordinates and manages storage, network and virtualization technologies to enable the dynamic placement of services on distributed infrastructures [Rahma et al(2013)]. OpenNebula is a pure private Cloud, where user can access Cloud functions by logging into the head node.

This interface is a wrapper around an XML-RPC interface, which can also be used directly [Sempolinski and Thain(2010)].

• Architecture: As shown in figure11, the architecture of OpenNebula [Hurwitz et al (2010)]. includes many components to handle the virtualized environment. These components are: OpenNebula core, scheduler and pluggable drivers that handle the external Cloud. To manage the life cycle of VM, the OpenNebula Core synchronizes storage, network and underlying a hypervisor. The Core performs specific storage, network or virtualization operation through pluggable Drivers. The Core is also designed to support deployment of services, which includes a set of inter related components (e.g. web server, DB backend, etc.) requiring several VMs. The core also provides different context information like IP address of the web server, digital certificates and software licenses to the VMs [Hurwitz et al (2010)].

The second important part is scheduler which schedules the VM. The OpenNebula default scheduler uses a rank scheduling policy that places VMs on physical resources according to a ranking algorithm that is highly configurable by the administrator, and uses a real-time data from both the running VMs and available physical resources to schedule VM. It proffers Management Interfaces to integrate the Core functionality such as accounting or monitoring frameworks within other data center management tools. OpenNebula implements the libvirt API and a command line interface (CLI) as an interface for VM management. OpenNebula can also support a hybrid Cloud model by using Cloud drivers to interface with external Clouds.

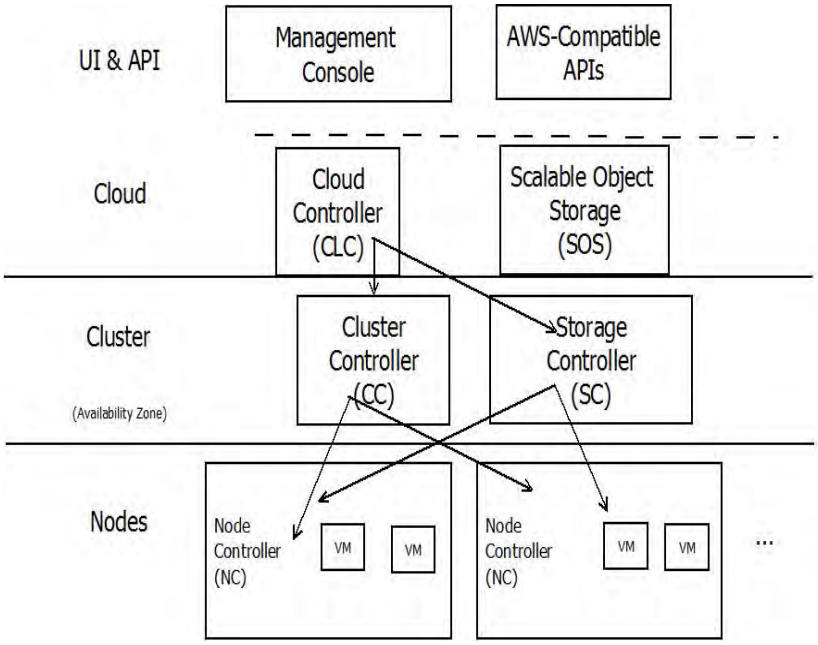

Figure 10: Eucalyptus [Sempolinski and Thainl(2010)]

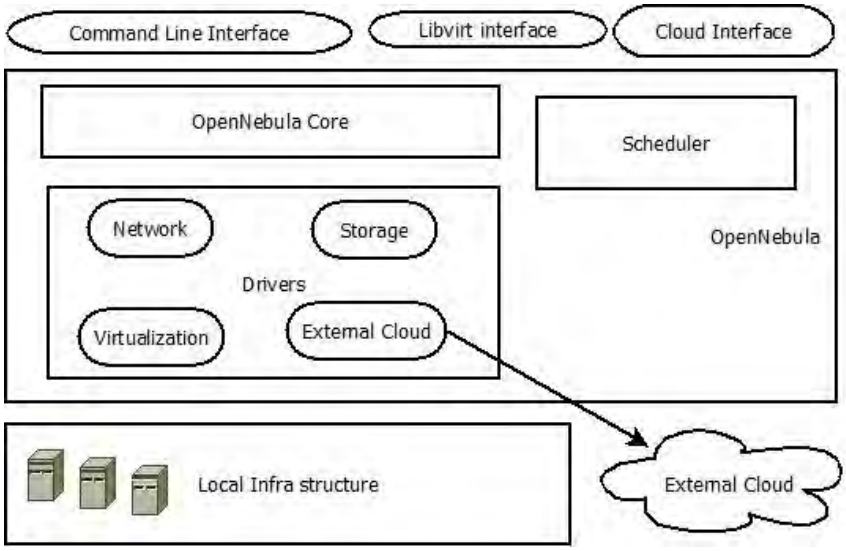

Figure 11: OpenNebula[Hurwitz et al (2010)].

3. Nimbus [Sefraoui et al(2012)]: The Nimbus advertises itself as a "science" Cloud solution [Sempolinski and Thain(2010)] It can be introduce as a combination of open source tools, which provides IaaS Cloud computing solutions to users, which provides a remote resources on lease by deploying VMs and configured VM according to user's requirement. It is formally known as Virtual Workspace Service (VMS). Generally workspace service is a technical component in the software collection. In nimbus most of the customization is done by the administrator and is not allowed to the users. The programming framework used for nimbus is Java and Python.

Architecture: As shown in figure 12, the architecture of Nimbus includes various components. The roles of each of these components are as follows:

Workspace service: Workspace service is the most important component of the architecture, which is responsible for management of virtual machines in data center. Moreover, it receives developers' requests through different interfaces. It uses the WSRF-based (Web Services Resource Framework 40) interface and the Amazon EC2-compatible as an interface.

• The workspace control (or VMM): The workspace control performs basic management and control operations of the VMs. So, it needs to be installed on each server.

• Workspace Resource Manager and Workspace Pilot: These components handle automatic management of VMs including selection of appropriate server for each VM.

• The ContextBroker: It allows clients to coordinate large virtual cluster launches automatically and repeatably.

• The ContextAgent: It lives on VMs and interacts with the Context Broker at VM boot.

4. OpenStack [Endo et al(2010)]: OpenStack is Scalable,Compatible, Flexible(i.e OpenStack supports most virtualization solutions of the market like ESX, Hyper-V,KVM, LXC, QEMU, UML, Xen and XenServer). It is written in python and implements EC2 and Rackspace as a control APIs. To provide an interface with a maximum number of hypervisors like Xen, KVM, HyperV and Qemu, it uses different drivers. Linux container technology such as LXC is also supported for situations where users wants to minimize virtualization overhead to achieve efficiency and sufficient performance. Moreover, to different hypervisors, OpenStack also supports ARM and many alternative hardware architectures[Borja et al (2009)].

Architecture: As shown in figure 13, the architecture of of OpenStack includes major three components known as OpenStack Compute, Image and Object. The roles of each of these components are as follows:

- OpenStack Compute: It is also known as Nova. It works as a management platform that controls the infrastructure. It provides an administrative interface and API for the proper coordination in Cloud. It does not require any prerequisite hardware/installation and it is independent to any hypervisor. It has a seven main components included:

(a) API Server: Which acts as a web front and service control of the hypervisor.

(b) The Message Queue: Message Queue implements the mechanism for sending the exchanged instructions for smooth communication.

(c) A Compute Controller: It controls the life cycle of instances and responsible of creating and managing virtual servers.

(d) The component Object Store : It provides storage services.

(e) A Volume Controller component: It controls the volumes.

(f) The network Controller: It is responsible for controlling the network. Lastly,

(g) Scheduler: It schedules tasks and allocate the virtual server.

- OpenStack Imaging Service: Imaging Service provides storage services. It also manages the distribution of the images to virtual machine disks.
- OpenStack Object Storage: It used to create a redundant and scalable storage space for storing multiple petabytes of data. It is used for long term storage of large volumes.

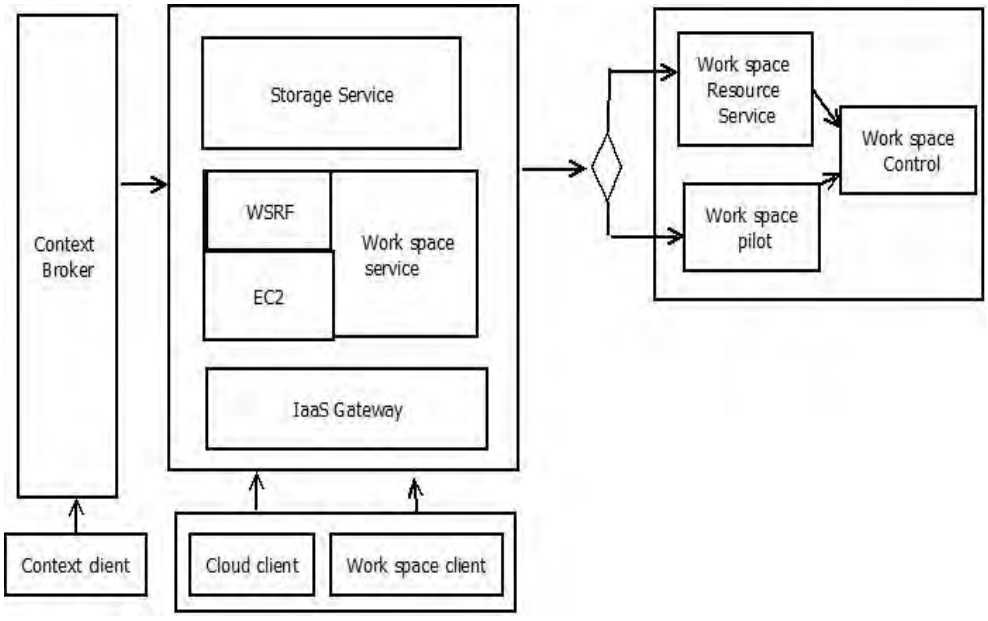

Figure 12: Nimbus[Sefraoui et al(2012)]

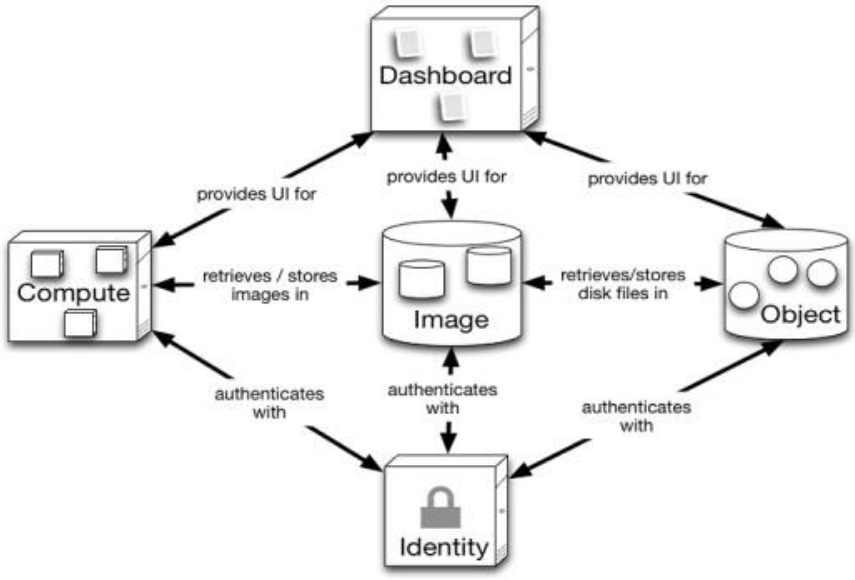

Figure 13: OpenStack [Borja et al (2009)]

5. CloudStack[Nurmi et al(2009)]: CloudStack is completely open source which offers Cloud including public, private and hybrid. It has Java based implementation. It is compatible with Xen, KVM, VMWare. CloudStack provides two interfaces: a Web interface and a RESTful API.

Architecture: As shown in figure 14, CloudStack has five types of components viz. Computer Nodes, Clusters, Pods, Availability Zones, and the Management Service. Explanations of these components are as follows [48]:

• Computer Nodes: The Computer Nodes (CNs) are the resources that have the CloudStack agent and any of the hypervisors supported by the platform like KVM, XenServer or VMware vSphere installed.

These nodes can be identified by the machine that allows the execution of VMs.

• Clusters: A group of CNs is a Cluster. CNs of the same Cluster have the same hypervisor installed and share the same primary storage.

• Pods: A collections of Clusters is called as a Pod.

• Availability Zones: Collection of Pod is called as Availability Zones.

• Management Service: Management Service manages the entire Cloud. It controls and manages the services of Cloud.

The CloudStack platform has three different users with the different roles:

(a) Root Administrator: The root administrator has the highest access privileges. It can manage the entire Cloud, including the hosts, clusters, Pod, user accounts, services offered and many more.

(b) Domain Administrator: The domain Administrator is responsible to perform administrative operations of only one domain, but cannot access physical servers or other domains.

(c) CloudStack user: The CloudStack user has no privileges. It can only manage and handle its own virtual resources like VMs.

6. Ganeti [Nagar and Suman(2014)]: Ganeti is a virtual machine cluster management tool developed by google. It is light weight, easy to implement for a small cluster for organizations. It is compatible with Xen, KVM, VMWare. It doesn't provide object/image storage by default and does not provide any direct interface to users.

   Architecture: As shown in figure 15, architecture of ganeti has two main components: cluster and node group.

   • Cluster: Collection of node is called cluster. Instances (guests) run on nodes. A cluster has one master node that can handle and control the all other nodes in cluster. It control and handle VM by VM migration in case of failure or overload in same cluster.

   • Node group: Collection of cluster is known as Node group which actually is a split up of nodes into logical groups.

Enomaly [50]: Enomaly is an open source tool providing IaaS service. It focuses on small Cloud implementation. It can be deployed with the Xen and KVM. Open source edition of Enomaly suffers from many restrictions like limited scalability, no mechanism for capacity control.

8 Apache VCL [Rahma et al(2013)]: Apache VCL is an open source tool for providing SaaS. It focuses on Applications of the internets. It can deploy with VMware. Apache VCL has a simple architecture consisting of three tiers:
Web server, Database server and Management nodes
• Web server: This Components represents the VCL portal and uses Linux/Apache/PHP solution. This portal uses to enable the requesting and management of VCL resources through user interface.
• Database server: It stores the information like VCL reservations, different access controls and machine and environment inventory. It uses Linux/SQL solution for server.
• Management nodes: It is the processing engine. This controls a subset of VCL resources like physical blade servers, traditional rack, or virtual machines. It uses Linux/VCLD (perl)/image library solution. VCLD is a middle ware responsible to process reservations or jobs assigned by the VCL web portal. According to type of environment requested, VCLD provides a service (computational environment) to available user. Figure 16 shows a conceptual overview of the VCL, where to access its resources through a web interface, the user must have to connect firstly to the VCL Scheduling Application. Users may request a reservation to use the environment immediately or schedule to use it in the future.

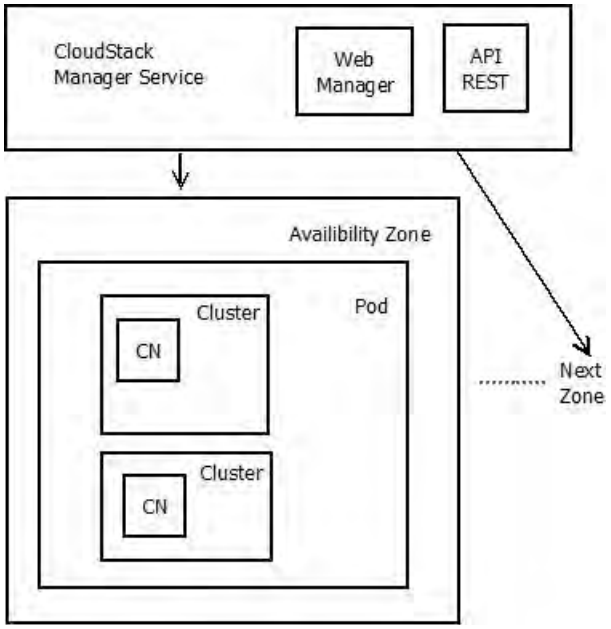

Figure 14: CloudStack[48]

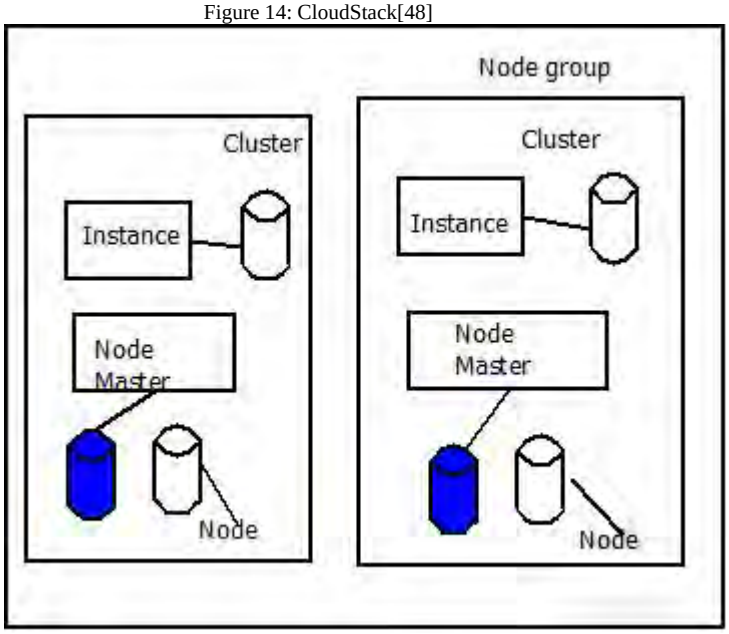

Figure 15: Ganeti[Muñoz et al (2012)]

9 TPlatform [Rahma et al (2013)]: TPlatform is used for providing services of PaaS services. It focuses on web mining.

Infrastructure of TPlatform is supported by three technologies viz. A scalable file system called Tianwang File System (TFS), the BigTable data storage mechanism and the MapReduce programming model.

Architecture [Gonçalves et al(2011)]: As shown in figure 17, architecture of TPlatform has two main components: PC cluster and Infrastructure and Data processing applications.

• PC Cluster: This layer is responsible for controlling the hardware infrastructure for data processing.

• Infrastructure: This layer includes different components of infrastructure like of file system (TFS), programming model (MapReduce) and distributed data storage mechanism (BigTable).

• Data Processing Applications: This layer facilitates the services for users to develop their application like web data analysis and language processing and many more.

XCP [Rahma et al (2013)]: It provides IaaS. It focuses on Automatic management in Cloud. It can be deployed in Xen. It cannot provide overall architecture of Cloud as it cannot provide any interface to end users for interaction.

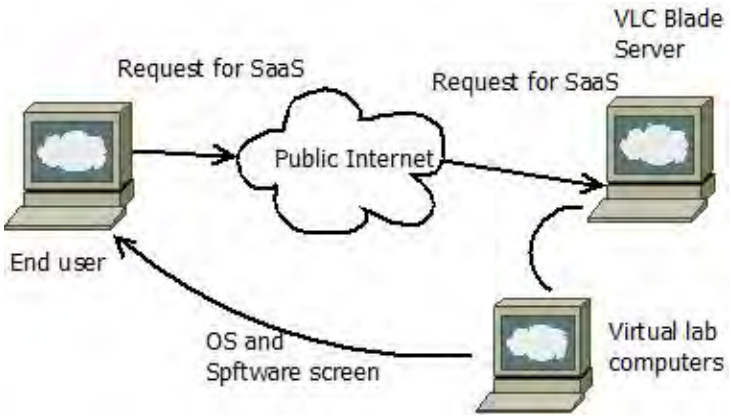

Figure 16: ApacheVCL [Gonçalves et al(2011)].

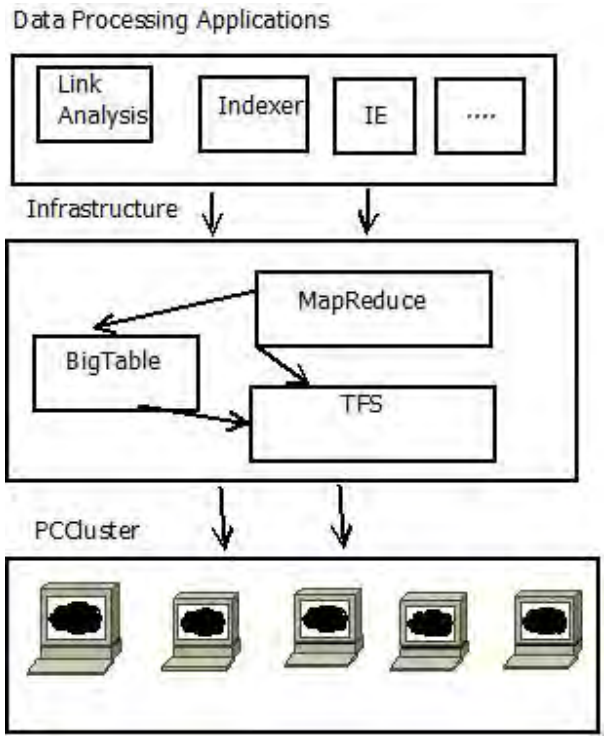

Figure 17: Tplatform[Gonçalves et al(2011)].

Discussion: The Eucalyptus Cloud is mainly used for private Clouds. While, OpenNebula can be considered as a "standard-based open-source toolkit to build private, public, and hybrid Clouds" and can be used with the Amazon EC2 service. VMM configuration set up is easy in Eucalyptus, as the details of VM are set by administrator for all users while its beings tedious in OpenNebula where all users must be made aware of some underlying configuration in order to properly configure VMs, as it requires users to provide more details of the VM. It has been analyzed that Nimbus emphasizes more towards science community. This may be less interested in the internal technique of the system. But, nimbus needs broad customization requirements. So, this

science community need be more familiar with the Globus Toolkit, while XCP is introduce to manage a virtualized infrastructure and does not offer a solution for resource's negotiation. OpenStack is excellent for large deployment, it has medium complexity for setup. It uses a python only for coding. Cloud Stack has lots of features. It also has medium complexity for set up as it has Monolithic component architecture. Ganeti has built-in fault tolerance. It is Ideal for small clusters. It is less complex, but is having few features. It has no EC2 compatibility. Summary of a discussion on Cloud architecture is shown in table2.

CONCLUSION AND FUTURE WORK

Cloud computing is identified as a new and most promising paradigm delivering IT services as computing utilities. As Clouds are designed to provide services on pay per use bases to end users where providers need to be recompense for sharing their resources and capabilities. In this paper, we have discussed various open source Cloud architectures and hypervisors. In particular, we have discussed hypervisors like KVM, Xen, LXD, Lguest and VirtualBox with container technologies like OpenVz, LXC. In the section of open-source Cloud architectures/software platforms, we covered OpenNebula, Eucalyptus, Nimbus, OpenStack, CloudStack, TPlatform, ApacheVCL, Ganeti and XCP. Moreover, we have discussed some representative platforms for Cloud computing covering the state-of-the-art. In analyzing these various open-source hypervisors and Cloud computing architecture/Cloud software, we found that there are salient differences between them in the overall scheme of their design and function. They identified to be useful based on the Cloud requirements for deployment. As the paper provides brief introduction of many open source IaaS platforms and architectures, it may be helpful to researchers/users to answer the question which open source IaaS platforms and architectures is right for them. Also, it gives a huge space to start a further work by selecting a best option for hypervisor and Cloud architectures/software available in open source. However, detail working of hypervisors /Cloud platforms and architectures are not explained/covered in present paper which can be covered in future work. Moreover, this Cloud architectures and platforms are not compared by performing any experiments on them. Only theoretical analysis is done. So, comparison of this open source Iaas frameworks based on experimental results can be targeted in future work.

Table 2: Comparison of Cloud architecture

Cloud archite	Cloud type	Hypervisor support	OSsup-ported	Supported languages	Main Char-	Service	Application
Eucalyptus	Private	VMWare,KVM,	Linux and Windows	Java and C	Hierarchical Architecture	IaaS	Large and Small
OpenNebula	Public, Privateand Hybrid	VMware, KVM, Virtual- Box,	Linux	C++, C, Ruby, Java, Shellscript, lex and	Policy-driven resource allocation	IaaS	Largeand Small Organi-zation.
CloudStack	Public, Privateand Hybrid	Xen,KVM, VMware	Linux	Java	Deploy and manage large	IaaS	largeand small organi-zations.
Nimbus	Private and Public	Xen 3.x or	Linux	Javaand Python	Legacyclus-ters for Cloud	IaaS	Small organi-zation.
OpenStack	Public, Privateand Hybrid	Xen,KVM, Hyper-VQEMU, UML(User	Linux	Python	Supports ARM and many altern	IaaS	Best option for Private and public Cloud
Ganeti	Private	Xen,KVM, VMWare	Linux	Python, Haskell,	lightweight, easy to	IaaS	Small organi-zation
Enomaly	Private	Xen,KVM andVirtual-	Linux	Python 2.5	Targeting small Cloud	IaaS	Small Organi-zation
Apache VCL	Private	VMware	Linux	PHP	Applications on the	SaaS	
TPlatform	Private	TFS and MapReduce	Linux	Python,C, Java, C++,R	Developme nt platform for	PaaS	Small organi-zation
XCP	Private	Xen	Windows and Linux	—	Only tool for automatic managemen t	IaaS	Useful envi-environmen for administra-tors and an

References

[1] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," Journal of internet services and applications, vol. 1, no. 1, pp. 7–18, 2010.
[2] B. Sosinskyl (2010)]. B. Sosinsky, Cloud computing bible. John Wiley & Sons, 2010, vol. 762.
[3] Jing et all (2013)]. S.-Y. Jing, S. Ali, K. She, and Y. Zhong, "State-of-the-art research study for green cloud computing," The Journal of Supercomputing, vol. 65, no. 1, pp. 445–468, 2013.
[4] Armbrust et al(2010)]. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica et al, "A view of cloud computing," Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010.
[5] [Online]. Available: http://aws.amazon.com/ec2
[Online]. Available: www.ibm.com/in/en/
[6] [Online].Available:https://cloud.google.com/appengine/?utm_source=google&utm_medium=cpc&utm_campaign=2015-q3-cloud-japac-in-gae-bkws-freetrial&utm_content=en&gclid=COvHrszd9ccCFQGSjgodIsUORw
[7] [Online]. Available: http://http://azure.microsoft.com/en-in/?WT.srch=1&WT.mc_id=SEM_1ncz5e1t
[8] [Younge et al(2011)]. A. J. Younge, R. Henschel, J. T. Brown, G. Von Laszewski, J. Qiu, and G. C. Fox, "Analysis of virtualization technologies for high performance computing environments," in Cloud Computing (CLOUD), 2011 IEEE International Conference on. IEEE, 2011, pp. 9–16.
[9] [Sabahi (2012)]. F. Sabahi, "Secure virtualization for cloud environment using hypervisor-based technology," Int. Journal of Machine Learning and Computing, vol. 2, no. 1, 2012.
[10] [Walters et ali (2008)]. J. P. Walters, V. Chaudhary, M. Cha, S. Guercio Jr, and S. Gallo, "A comparison of virtualization
[11] technologies for hpc," in Advanced Information Networking and Applications, 2008. AINA 2008. 22nd
[12] International Conference on. IEEE, 2008, pp. 861–868.
[13] [Mishra and Jaiswali (2012)]. R. Mishra and A. Jaiswal, "Ant colony optimization: A solution of load balancing in cloud," International Journal of Web & Semantic Technology (IJWesT), vol. 3, no. 2, pp. 33–50, 2012.
[14] [Abels et al (2005)]. T. Abels, P. Dhawan, and B. Chandrasekaran, "An overview of xen virtualization," Dell Power Solutions, vol. 8, pp. 109–111, 2005. Pg.17
[15] [Fenn et al (2009)]. M. Fenn, M. A. Murphy, and S. Goasguen, "A study of a kvm-based cluster for grid computing," in
[16] Proceedings of the 47th Annual Southeast Regional Conference. ACM, 2009, p. 34.
[17] [Sempolinski and Thainl(2010)] P. Sempolinski and D. Thain, "A comparison and critique of eucalyptus, opennebula and nimbus," in Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on. Ieee, 2010, pp. 417–426.
[18] [Soltesz et all (2007)]. S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, "Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors," in ACM SIGOPS Operating Systems Review, vol. 41, no. 3. ACM, 2007, pp. 275–287.
[19] [Deshane et al (2008)]. T. Deshane, Z. Shepherd, J. Matthews, M. Ben-Yehuda, A. Shah, and B. Rao, "Quantitative comparison of xen and kvm," Xen Summit, Boston, MA, USA, pp. 1–2, 2008.
[20] [Kivity et al (2007)]. A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: the linux virtual machine monitor," in Proceedings of the Linux Symposium, vol. 1, 2007, pp. 225–230.
[21] [Barham et al (2003)]. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," ACM SIGOPS Operating Systems Review, vol. 37, no. 5, pp. 164–177, 2003.
[22] [Cherkasova and Gardner et al (2005)]. L. Cherkasova and R. Gardner, "Measuring cpu overhead for i/o processing in the xen virtual machine monitor." in USENIX Annual Technical Conference, General Track, vol. 50, 2005.
[23] [Cherkasova et al (2007)]. L. Cherkasova, D. Gupta, and A. Vahdat, "Comparison of the three cpu schedulers in xen," SIGMETRICS Performance Evaluation Review, vol. 35, no. 2, pp. 42–51, 2007.
[24] [Kiszka et al (2010)]. J. Kiszka, C. T DE IT, and C. C. C. E. Linux, "Architecture of the kernel-based virtual machine (kvm)," in Technical Sessions of Linux Kongress, 2010.
[25] [Franssens et al (2013)]. N. Franssens, J. Broeckhove, and P. Hellinckx, "Taxonomy of real-time hypervisors," in P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on. IEEE, 2013, pp. 492–496.
[26] [Gu and Zhao (2012)]. Z. Gu and Q. Zhao, "A state-of-the-art survey on real-time issues in embedded systems virtualization," Journal of Software Engineering and Applications, vol. 5, no. 4, 2012.
[27] [Van Doorn(2006)]. L. Van Doorn, "Hardware virtualization trends," in ACM/Usenix International Conference On Virtual Execution Environments: Proceedings of the 2 nd international conference on Virtual execution environments, vol. 14, no. 16, 2006, pp. 45–45.
[28] [Online].Available:
https://www.ibm.com/developerworks/community/blogs/ibmvirtualization/entry/kvm_myths_uncovering_the_truth_about_the_open_source_hypervisor?lang=en
[29] [Zaidenberg(2012)]. N. J. Zaidenberg, "Applications of virtualization in systems design," Jyväskylä studies in computing 153, 2012.
[30] [Online]. Available: https://www.virtualbox.org/
[31] [Fuertes and Vergara(2007)]. W. M. Fuertes and J. E. L. de Vergara, "A quantitative comparison of virtual network environments based on performance measurements," in Proceedings of the 14th HP Software University Association Workshop, Garching, Munich, Germany, 2007, pp. 8–11.
[32] [P. Li(2010)]. P. Li, "Selecting and using virtualization solutions: our experiences with vmware and virtualbox," Journal of Computing Sciences in Colleges, vol. 25, no. 3, pp. 11–17, 2010.
[33] [Kovari and Dukani(2012)]. A. Kovári and P. Dukan, "Kvm & openvz virtualization based iaas open source cloud virtualization
[34] platforms: Opennode, proxmox ve," in Intelligent Systems and Informatics (SISY), 2012 IEEE 10th Jubilee International Symposium on. IEEE, 2012, pp. 335–339.
[35] [Bazargan et al(2012)]. F. Bazargan, C. Y. Yeun, and M. J. Zemerly, "State-of-the-art of virtualization, its security threats and deployment models," International Journal for Information Security Research (IJISR), vol. 2, no. 3/4, pp. 335–343, 2012.
[36] [Online]. Available: https://linuxcontainers.org/
[37] [Online]. Available: http://www.novell.com/feeds/nih/wp-content/uploads/2012/05/SUS15_lec.pdf
[38] [Online]. Available: http://www.ubuntu.com/cloud/tools/lxd

[39] [Rahma et al(2013)]. F. Rahma, T. B. Adji, and W. Widyawan, "Scalability analysis of kvm-based private cloud for iaas,"
[40] International Journal of Cloud Computing and Services Science, vol. 2, no. 4, p. 288, 2013.
[41] [Gonçalves et al(2011)]. G. E. Gonçalves, P. T. Endo, T. Cordeiro, A. Palhares, D. Sadok, J. Kelner, B. Melander, and J. Mangs, "Resource allocation in clouds: concepts, tools and research challenges," XXIX SBRC-Gramado-RS, 2011.
[42] [Endo et al(2010)]. P. T. Endo, G. E. Gonçalves, J. Kelner, and D. Sadok, "A survey on open-source cloud computing solutions," in Brazilian Symposium on Computer Networks and Distributed Systems, 2010.
[43] [Sefraoui et al(2012)]. O. Sefraoui, M. Aissaoui, and M. Eleuldj, "Openstack: toward an open-source solution for cloud computing," International Journal of Computer Applications, vol. 55, no. 3, pp. 38–42, 2012.
[44] [Keahey (2009)]. K. Keahey, "Nimbus: open source infrastructure-as-a-service cloud computing software," in Workshop on adapting applications and computing services to multi-core and virtualization, CERN, Switzerland, 2009.
[45] [Nagar and Suman(2014)]. N. Nagar and U. Suman, "Architectural comparison and implementation of cloud tools and technologies," in Proc. of 4 th IEEE International Conference on Electronics Computer Technology (ICECT'12), Kanyakumary, India, IEEE Explore, ISBN, 2014, pp. 978–1.
[46] [Kumar et al (2014)]. R. Kumar, S. Agarwal, M. Bansal, and A. Mishra, "Open source virtualization management using ganeti platform," in National Conference on Emerging Technologies in Computer Engineering (NCETCE)–2014, Supported by: Computer Society Chapter, IEEE Delhi Section, 2014.
[47] [Peng et al (2009)]. J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang, and Q. Li, "Comparison of several cloud computing
[48] platforms," in Information Science and Engineering (ISISE), 2009 Second International Symposium on.
[49] IEEE, 2009, pp. 23–27.
[50] [Nurmi et al (2009)]. D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The
[51] eucalyptus open-source cloud-computing system," in Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on. IEEE, 2009, pp. 124–131.
[52] [Kranasi et al (2012)]. P. Kranas, V. Anagnostopoulos, A. Menychtas, and T. Varvarigou, "Elaas: An innovative elasticity as a service framework for dynamic management across the cloud stack layers," in Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on. IEEE, 2012, pp. 1042–1049.
[53] [Hurwitz et al (2010)]. J. Hurwitz, R. Bloor, M. Kaufman, and F. Halper, Cloud computing for dummies. John Wiley & Sons, 2010.
[54] [Borja et al (2009)]. S. Borja, M. Ruben, and M. Ignacio, "An open source solution for virtual infrastructure management in private and hybrid clouds," IEEE Internet Computing, vol. 1, pp. 14–22, 2009.
[55] [48] [Online]. Available: https://www.openstack.org/software/openstack-compute/
[56] [Muñoz et al (2012)]. V. M. Muñoz, V. F. Albor, R. G. Diaz, A. C. Ramo, T. F. Pena, G. M. Arévalo, and J. J. S. Silva, "The integration of cloudstack and occi/opennebula with dirac," in Journal of Physics: Conference Series, vol. 396, no. 3. IOP Publishing, 2012, p. 032075.
[57] [Online]. Available: http://events.linuxfoundation.org/sites/events/files/slides/openstack-vs-ganeti.pdf
[58] [Online]. Available: http://wiki.libvirt.org/page/Enomaly_Elastic_Computing_Platform_(ECP)