ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 5, Issue 4, April 2018

A Comparative Analysis between Traditional Virtualization and Container Based Virtualization for Advancement of Open Stack Cloud

[1] Krishna Parihar, [2] Arjun Choudhary, [3] Vishrant Ojha

[1][2] Department of Computer Science and Engineering, Sardar Patel University of police, Security and Criminal Justice, Jodhpur

Abstract: - **LXD based machine containerized new concept in the virtualization world. Its give performance to the bare metal virtualizations. A Centralized system is accessed through IP and can be scalable on demand as per requirement, Technology that eases the user problem like hardware, as well as software most popular technologies, is known as Cloud Computing. As per enterprise and industries demand cloud computing holds a big part of IT world. An OpenStack is a way to manage public and private cloud computing platform. A Linux Container Hypervisor brings Cloud Computing to the next generation Hypervisor with its features Fast, Simple, Secure. The main goal of this paper is factual calibrate is determine various virtualization technique and performance analysis.**

Keywords— **OpenStack, LXD, Cloud Computing, Virtualization, Container, Hypervisor**

I. INTRODUCTION

Analysis of structured and consistent data has seen A cloud computing technologies have been rapidly developing. resources are dynamically enlarged, virtualized as well as feed as a service over the Internet, it permits providers to provide users connected to a virtually unlimited number of resources i.e. Resource Outsourcing.cloud system is required all the field like education, medical industries, and many more place. Cloud Computing providing essential service i.e. infrastructure as a service (IaaS), network as a service (NaaS), platform as a service (PaaS), software as a service (SaaS). These all are made possible through virtualization.

A. Virtualization

Virtualization is the optimum way to enhance resource utilization in efficient manner. It refers to act of creating a virtual (similar to actual) variations of the system. Physical hardware is managed with the help of software and converted into the logical resource that will be in a shared pool or can be used by the privileged user.[1] This service is known as VMs we can say Infrastructure as a service. virtualization is base of any public and private cloud development. Most of the public cloud providers such as Amazon EC2, Google Compute Engine and Microsoft Azure leverage virtualization technologies to power their public cloud infrastructure.[2] The core component of virtualization is Hypervisors.

B. Hypervisor

It is a software which provides isolation for virtual machines running on top of physical hosts. The thin layer of software that typically provides capabilities to virtual parti-tioning that runs directly on hardware, It provides a potential for virtual partitioning and responsible for running multiple kernels on top of the physical host. This feature makes the application and process isolation very expensive. There will be a big impact if computer resources can be used more efficiently. The most popular hypervisors today are VMware, KVM, Xen, and Hyper-V. [4]

II. VIRTUALIZATION MODELS

The reference models on which the virtualization is based on Hypervisor for different types of cloud services.[5].
Type 1(Bare Metall)
Type 2(Hosted Model)

A. Bare Metall Virtualization

Hypervisors run directly on the system hardware. They are often referred to as a native or bare metal or embedded hypervisors in vendor literature.

B. Hosted Virtualization

This kind of virtualization service approach framed on top of a standard Host operating system (which is generally called host OS). By this approach, the virtualization software pro-vides the provision to talk directly to the underlying hardware. Both kernel and operating system take a while to boot. I/O performance is less efficient as compared to working with the single dedicated system. In order for the guest application to perform I/O operation, it needs first to call the guest kernel, which makes a request to what it considers with the hardware. Which is in turn emulated by the hypervisor, and passed to the host operating system, and finally to the hardware. The response is then passed the same circuitous route. Every single kernel gets its own perpetual memory footprint and bears a CPU overhead, so the overhead of virtualization is quite large.

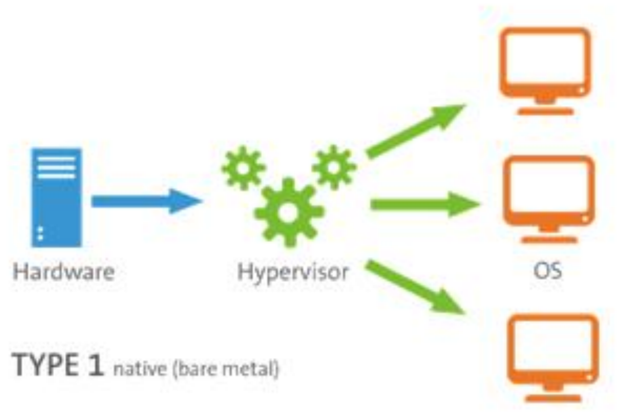


Fig. 1: Bare Matel

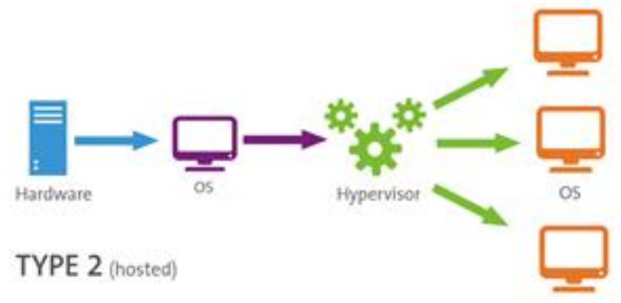


Fig. 2: Hosted Hypervisor

III. LXD

Recently researchers have focused on container-based vir-tualization such as LXC Linux container. Linux containers such as LXC have recently been developed. In container-based virtualization, a new container is dynamically built an interface on the same operating system (OS) within fractions of a second. A container is a virtual environment or replication of the running OS. Container-based virtualization is a lightweight virtualization method. A container is isolated and protected be-cause it cannot access another container. By using containers, we provide a secure virtual environment to applications. Linux is typically used as the OS in cloud instance in the container based virtualization consumes smaller resources than hardware virtualization. Booting is fast because there is no kernel to start up, in fact, even machine containers tend to carry lightweight operating systems with sub-second boot times. We can say it the future generation hypervisor for Linux. LXD perform a task with the speed and latency of containers and brings them to the new generation of hypervisor world. An LXD container gives you complete machine system features, not only a single processor thread, we can optimize that all applications can be installed in LXD containers without any need of touch the application to make it run because LXDs machine containers operate just like VMs. Easy monitoring of guest processes, take Snapshots, we can easily live to migrate and run Docker inside LXD.

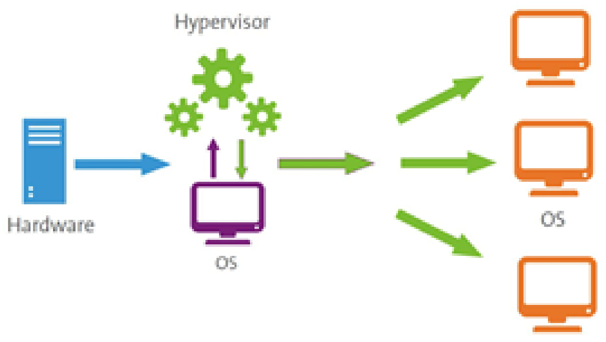


Fig. 3: Container Hypervisor

IV. CONTAINER

A container can be defined as a single operating system image, bundling a set of isolated application and their de-pendent resources so that they run separated from the host machine. There are multiple such containers within the same host machine. The generic term in the place of Linux universe, container offers more confines for

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 4, April 2018

confusion. Basically, a con-tainer is nothing but more than a virtual file system which are isolated with some Linux kernel features, such as namespaces and process groups, from the main physical system. Through containers framework it offers an environment as close as desirable one as we want from a VM but without the overhead that comes with running on an another kernel and simulating all the hardware.[7] Containerization differs from traditional virtualization technologies and offers many advantages over traditional virtualization Containers are lightweighted compared to traditional virtual machines. Containers share resources with the underlying host machine, with user space and process isolations. Due to lightweight nature of containers, more contain-ers can run per host than virtual machines per host. Starting a container happens nearly instantly compared to the slower boot process of virtual machines. Containers are portable and can reliably regenerate a system environment with required software packages, irrespective of the underlying host operating system. Unlike containers, virtual machine require emulation layers (either software or hardware), which consume more resources and add additional overhead.

A. Features to Enable LXD Containers

Containers rely on the following features in the Linux kernel to get a contained or isolated area within the host machine. This area is closely related to a virtual machine but without the need for a hypervisor.[9]

1. Simple:- Easy sharing of a hardware resources clean command line interface, simple REST API. 2. Fast:- Rapid provisioning, instant guest boot, and no virtualization overhead so as fast as bare metal. 3.Secure:- Secure by default, combine all available kernel security feature with AppArmor, user namespaces, SECCOMP. 4. Scalable:- Very precise quality-of-service from a single container on a developer laptop to thousands container per host in a datacentre. Remote image services with Extensible storage and networking. 5. Control groups (cgroups) :- it is a kernel mechanism for grouping and tracking. the kernel-provided administration interface is through a virtual filesystem. its goal is to respond to dbus requests from any user. 6. Namespaces:- A namespace (ab-breviated as NS) produce a confined instance of the global resource. Basically Used for implementation of containers. A lightweight virtualization that gives us processes illusion that they are the only instance running in the system. 7. AppArmor:- It is a Mandatory Access Control (or MAC) mechanism. It is a security system that keeps applications from turning interrupted. It uses LSM kernel for improvement the restrict programs to certain resources. when the system starts, AppArmor loads profiles into a kernel. Complain and enforcement are the profile modes for AppArmor.

B. Performance

Density:- in density test Lxd launch 536 guests and every single guest was full ubuntu system whereas KVM just launch 37 guests, which are nearly 15result, there is impressive growth in no of instances which can be wrapped into a single machine that give compelling betterment of the cost by the optimal resource utilization.[10]

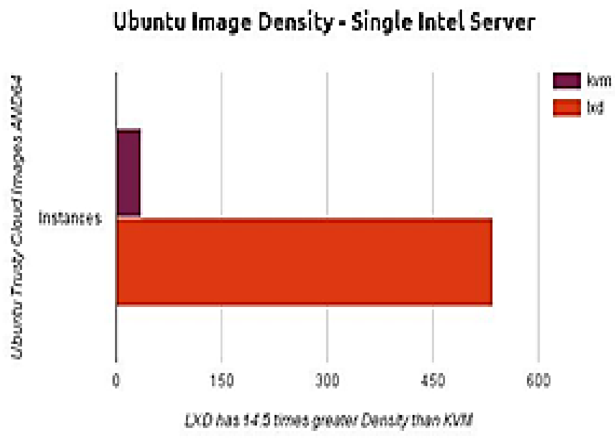


Fig. 4: density Analysis

Speed:- there is a high difference in startup time between the two technologies. LXD guests start in 1.5 seconds While KVM took 25 seconds to start and launch.

Latency:- LXD container run critical applications at a bare metal performance. without emulation of a virtual machine, Its skip the scheduling bulky performance hazards and latencies which generally, we faced in virtualization. According to Canonical testing report compared with KVM, LXD has 57less latency for guests.

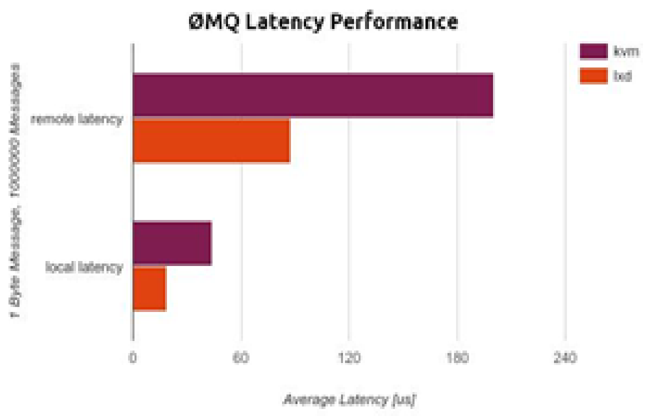
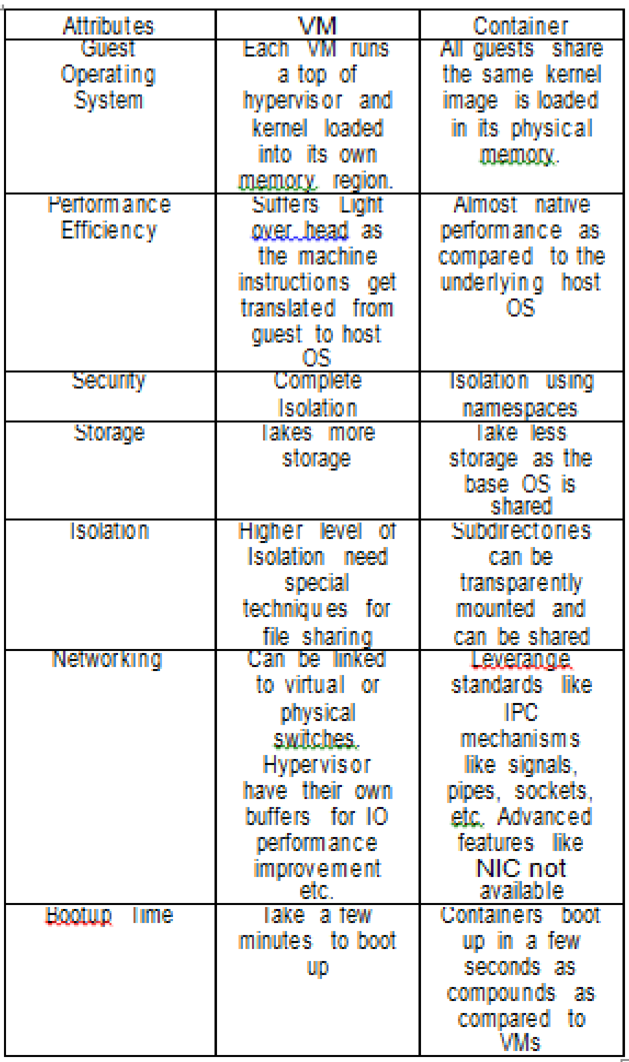


Fig. 5: Speed Analysis

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 5, Issue 4, April 2018



Attributes	VM	Container
Guest Operating System	Each VM runs a top of hypervisor and kernel loaded into its own memory region.	All guests share the same kernel image is loaded in its physical memory.
Performance Efficiency	Suffers Light over head as the machine instructions get translated from guest to host OS	Almost native performance as compared to the underlying host OS
Security	Complete Isolation	Isolation using namespaces
Storage	Takes more storage	Take less storage as the base OS is shared
Isolation	Higher level of Isolation need special techniques for file sharing	Subdirectories can be transparently mounted and can be shared
Networking	Can be linked to virtual or physical switches. Hypervisor have their own buffers for IO performance improvement etc.	Leverange standards like IPC mechanisms like signals, pipes, sockets, etc. Advanced features like NIC not available
Bootup Time	Take a few minutes to boot up	Containers boot up in a few seconds as compounds as compared to VMs

Table 1: Difference Between VM and Container

V. CONCLUSION

As seen in the paper above we can say that LXD is new kind of hypervisor which brings Cloud Computing to the next generation. LXD gives optimal performance. We can use LXD Machine Container to deploy OpenStack private cloud and manage all the components efficiently. because it is similar to Hosted Hypervisor virtualization but it gives us Bare Metal performance. We observe its performance by several statistical analysis like process latency, instance launch etc. we compare these all performance with another type of virtualization which is popular in the market and most frequently uses. LXD has kernel level security by default. It also suitable for Docker instance run within Machine Container with more efficiently. LXD is the solution for traditional virtualization technique.

REFERENCES

[1] Charalampos Gavriil Kominos and Nicolas Seyvet, Bare-metal, Virtual Machines and Containers in OpenStack , Uppsala, Sweden, 2014.

[2] Minoru Uehara, Performance Evaluations of LXC based Educational Cloud in a Bare Metal Server , Saitama, Japan, 2017.

[3] Prof. Ann Mary Joy, Performance Comparison Between Linux Containers and Virtual Machines, Ghaziabad, India, 2015.

[4] David Bernstein, Containers and Cloud: From LXC to Docker to Kuber-netes, Cloud Computing, 2014.

[5] Roberto Morabito, and Jimmy Kjllman Hypervisors vs. Lightweight Virtualization: a Performance Comparison, Jorvas, Finland 2016.

[6] Alan D Waite Adding Speed and Agility to Virtualized Infrastructure with OpenStack, , 2015.

[7] Kuniyasu Suzaki, Hidetaka Koie and Ryousei Takano Bare-Metal Con-tainer, SmartCity-DSS, 2016.

[8] Sapan Gupta, and Deepanshu Gera A Comparison of LXD, Docker andVirtual Machine, 2016.

[9]enthil Kumaran S. Practical LXC and LXD, Tamil Nadu, India 2016.

[10]By Canonical LXD crush KVM in density and speed, 2015. https://insights.ubuntu.com/2015/05/18/lxd-crushes-kvm-in-density-and- speed/

[11] Rakesh Kumar, Neha Gupta, Shilpi Charu, Kanishk Jain, and Sunil Kumar Jangir Open Source Solution for Cloud Computing Platform Using OpenStack, Jaipur, India, 2014.

[12] Container retrievedon September 15, 2015 from https://www.ubuntu.com/containers

[13] The no-nonsense way to accelerate your business with containers retrieved on September 9, 2017 from https://pages.ubuntu.com/rs/066-EOV-

335/images/WPT henononsense way to accelerate your business withcontainers:pdf

[14] LXD Introduction retrieved on Octomber 10, 2017 from https://linuxcontainers.org/lxd/introduction/

[15] LXD container hypervisor retrieved on Octomber 7, 2017 from https://www.ubuntu.com/containers/lxd

[16] LXD crushes KVM in density and speed, Article by Canonical. Retrieved on Octomber 25, 2017 from https://insights.ubuntu.com/2015/05/18/lxd-crushes-kvm-in-density- and-speed/

[17] Adding Speed and Agility to Virtualized Infrastructure with OpenStack retrieved on November 25, 2017 from https://www.openstack.org/assets/pdf downloads/virtualization- Integration-whitepaper-2015.pdf

[18] Practical LXC and LXD Linux Containers for Virtualization And Orchestration retrievedon November 20, 2017 from https://link.springer.com/book/10.1007/978-1-4842-3024-4