Volume 4, Issue 1, January 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Performance Analysis of Linux Containers - An Alternative Approach to Virtual Machines

Sudha M*, Harish G M, Usha J
Department of MCA,
R V College of Engineering, India

Abstract— Rapid advancement in computing technology has enabled Cloud computing to be a paramount paradigm in distributed systems. It is essential to fully understand the underlying technologies that make clouds possible. One key technology that make makes the cloud popular is virtualization. Even though virtualization technology is not new, the concept of hypervisors with virtualization is getting popular and also well understood by many. In this paper we discuss about two different virtualization technologies and investigate the performance of Kernel based Virtual Machine and Linux Containers.

Keywords— Linux Containers, Virtual Machines, Hypervisor, Virtualization.

I. INTRODUCTION

In the recent years, virtualization technology has become very popular and is gaining momentum with cloud computing. Software solutions such as xen, vmware, kvm with the incorporation of hardware support in processors viz, intel-vt and amd-v has given a giant leap to cloud computing. The key benefits of virtualization include hardware independence, availability, isolation and security. However, virtualization consumes overhead in terms of I/O. Recent operating container-based virtualization implementations, viz Linux-VServer, OpenVZ, Linux Containers (LXC), offer lightweight virtualization layer, which promises a near-native performance. With technological advancements, CPUs with Multi-cores are the de-facto standard. Every single core in a multi-core CPU has the power to run a physical machine. This can be achieved through virtualization. Even though the concept of virtualization is not new, recent demands, such as maximum hardware utilization, reduced hardware costs, optimal power consumption and on-demand access have led to significant increase in the deployment of virtualization and the number of available virtualization solutions. It enables multiple and varied operating system instances to run at the same time on a physical computer system, with each instance sharing the physical resources of the host computer system, such as memory, CPU, etc[1]. A virtual machine(VM) is a "completely isolated guest operating system installation within a normal host operating system"[2].

II. VIRTUALIZATION AND VIRTUAL MACHINES

Popek and Goldberg were the first to define virtual machines as "an efficient, isolated duplicate of a real machine". Presently, VM which have no direct interaction to any of the real hardware is popular[3[13]]. A virtual machine(VM) is a software that emulates physical machine. They are classified into two categories based on their use and degree of correspondence to any real machine as Process Virtual Machine and System Virtual Machine.

A. Process Virtual Machines

A Process Virtual Machine (PVM) or application VM is designed to run a single program with a single process. It runs as a normal application inside a host OS and supports a single process. The VM is created when a process is initiated and destroyed when the process exits or dies. PVM is also referred to as application virtual machine. It runs as a normal application inside a host OS and supports a single process. This VM mainly aims at providing a platform-independent development environment. Java programming language is platform independent as it implements Java Virtual Machine (JVM) which is a process VM.

B. System Virtual Machines

A System Virtual Machine gives a complete virtual hardware platform with support for execution of a complete operating system (OS).

Advantages[4]:

- **M**ultiple Operating System environments can run in parallel on the same piece of hardware in strong isolation from each other.
- The VM can provide an instruction set architecture (ISA) that is slightly different from that of the real machine,

Disadvantages:

- Since the VM indirectly accesses the same hardware the efficiency is compromised.

- Multiply VMs running in parallel on the same physical machine may result in varied performance depending on the workload imposed on the system. Implementing proper isolation techniques may address this drawback.

In the next section we discuss the classification of Virtualization techniques.

C. Virtualization Techniques :

There are many virtualization techniques, namely, hardware or platform virtualization, full virtualization, partial virtualization, para virtualization, network virtualization, application virtualization etc., Each technique has its benefits. Hardware or platform virtualization aims at creating a VM that behaves or performs just like a real machine with its own operating system. Applications that run on these VM are independent of the resources of the underlying hardware. For example, a computer running Microsoft Windows Operating System may host a virtual machine that runs a Linux Operating system like Ubuntu[5][6]. The software that creates Virtual Machines on the host is known as Hypervisor. In Full Virtualization the virtual machine is the exact simulation of the physical machine which allows the guest operating system to run unmodified. Partial Virtualization simulates few of the resources of the actual hardware. Hence, some programs on the guest need to be modified for execution in this environment. In para virtualization, a hardware environment is not simulated, however the guest programs are executed in their own isolated domains, as if they are running on a separate system. Guest programs need to be specifically modified to run in this environment. Network Virtualization provides a virtual network environment. Ex.VLANs.The abstraction provided by VLANs has made it feasible for companies to partition physical connections to define and create less expensive and flexible networks to meet the ongoing business demands [6][11]. Application virtualization provides an abstraction in which the traditional applications are wrapped inside a container, thus giving an illusion to the application that it is running on an intended platform. The application considers that it has access to the resources that it needs for execution of the task.

III. LINUX CONTAINERS

Linux Containers are better known as LXC are an operating system-level virtualization method which can be used for running multiple isolated servers (containers) on a single control host. LXC does not create a virtual machine, but provides a virtual environment with its own processes and network space. It is very similar to chroot, but provides more isolation [8]. LXCs are turning out to be a useful tool to run a variety of applications in isolated environments.

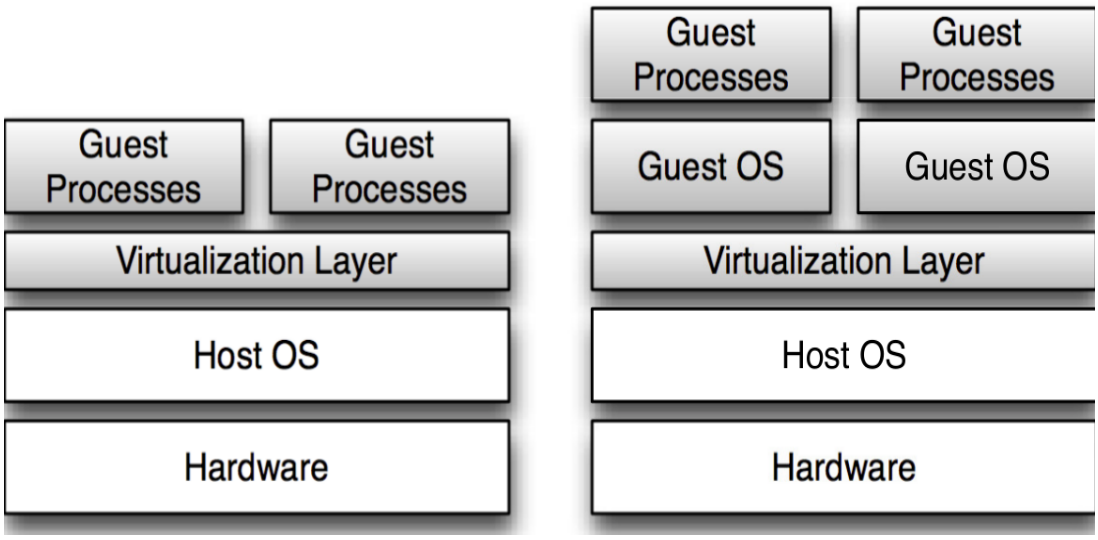

Fig. 1 : Comparison of container-based and hypervisor-based virtualization

LXC is based on various features of the Linux kernel such as chroot, PID and Network namespaces, etc. LXC is a user space interface for the Linux kernel container features. It allows Linux users to easily create and manage system application containers through its simple APIs[12][15]. LXCs aim to create an environment close to standard Linux installation by eliminating the need for a separate kernel. Figure 1 depicts both the virtualization technologies. The container based virtualization works at the operating system level and thus all virtual instances share a single operating system kernel. This reason makes it possible for container based virtualization to have a weaker isolation when compared to hypervisor based virtualization. However, from the user perspective, each container looks and executes exactly like a standalone OS [9][14]. The isolation in container-based virtualization is done by kernel namespaces. This feature of the Linux kernel allows different processes to have different view of the system. Since containers should not interact with things outside, many global resources are wrapped in namespace layer that provides the illusion that the container is its own system [10].

IV. METHODOLOGY

We conducted a computation and I/O performance test using Apache Benchmark and IO zone File system Benchmark tools to compare the performance of the containers with hypervisor-based virtualization with that of a host machine performance and check which gives a near bare metal performance. Table 1 depicts the details of the system used for the experiment.

TABLE I
SYSTEM CONFIGURATION

System	Configuration
ubuntubase [the Host Operating System]	Processor: Intel Core i3-2330M @ 2.20GHz (4 Cores), Motherboard: LENOVO 3254AM4, Memory: 2048MB, Disk: 500GB HITACHI HTS72505, OS:Ubuntu 12.04, Kernel: 3.10.5-031005-generic (i686), File-System: ext4,
ubuntuLxcBase [LXC container]	Processor: Intel Core i3-2330M @ 2.20GHz (4 Cores), Motherboard: LENOVO 3254AM4, Memory: 2048MB, Disk: 500GB HITACHI HTS72505, File-System: ext4
ubuntuKvmBase [Hypervisor]	Processor: QEMU Virtual 1.0 @ 2.20GHz (4 Cores), Motherboard: Bochs, Chipset: Red Hat Virtio, Memory: 1024MB, Disk: 7GB, OS: Ubuntu 12.04, Kernel: 3.2.0-23-generic-pae (i686), File-System: ext4, System Layer: QEMU 1.0

V. OBSERVATIONS AND DISCUSSIONS

A. Experiment 1: Computational Benchmark Result

The benchmarking was done using Phoronix Test Suite. Apache The aim was to compare the performance of Hypervisor - KVM and LXC. We configured VM and the container and used Phoronix Test Suite which comes bundled with test cases. We used "Timed Apache Compilation" test case bundled with Phoronix test suite. This test will determine the time taken to build the Apache HTTP Server and analyze the performance of the VM and container Phoronix Test Suite is a standard open source benchmarking tool for analyzing Virtual Machines Performance [10]. This is an open-source software licensed under the GNU GPLv3. It is the most comprehensive benchmarking and testing tool available with an extensible framework with QAflexibility ease in adding test cases. This software will effectively carry out quantitative and qualitative benchmarks.source code was compiled on all the three test systems. Figure 2, depicts the results obtained from Phoronics Test Suite. The chart represents the requests made per second during compilation. We observe that greater the number of requests the better is the performance. It was also observed that the performance of lxc container was closer to the host operating system performance than that of the kvm (hypervisor based Virtual Machine). The observations are as follows : The base system performance was 9708.01 requests. While KVM resulted in 7124.55 requests per second, LXC resulted in 7600.65 requests per second.

B. Experiment 2: IO Performance Comparison Result

The benchmarking was done using IO zone File System Benchmarking Tool. Figure 3. depicts the results produced by Iozone File System Bench Marking. The graph represents random IO requests per second and size of the fetched data. We observe that as the size of record increase the performance also increases. It is observed that the performance of LXC container was better than hypervisor based Virtual Machine-KVM.

The write and read performance of KVM was found better when the record size is small and as the record size increases the performance decreases. LXC performance is better when the record size is larger The write and read performance is depicted in Fig.3 and Fig. 4 respectively.

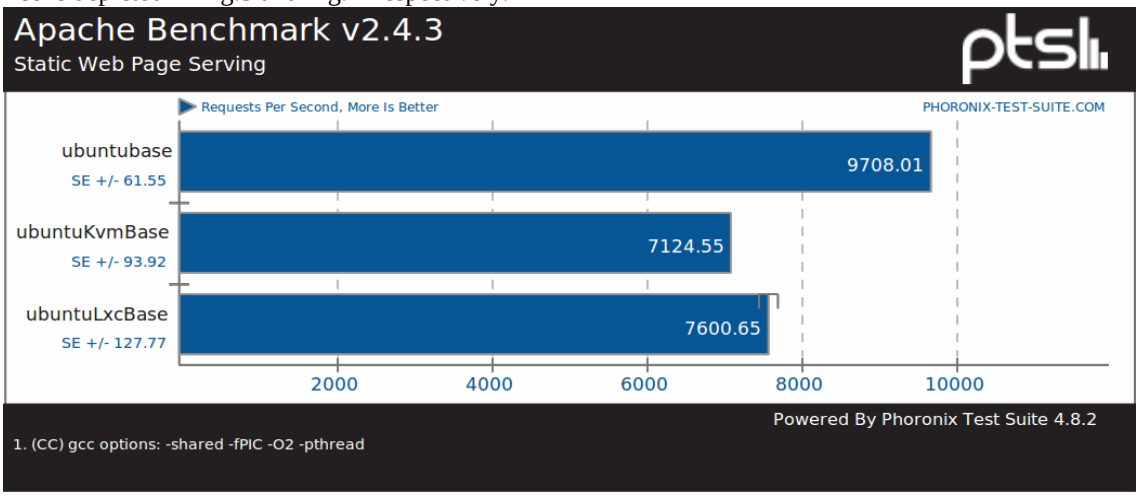

Fig 2. Apache Source Compilation Test Result

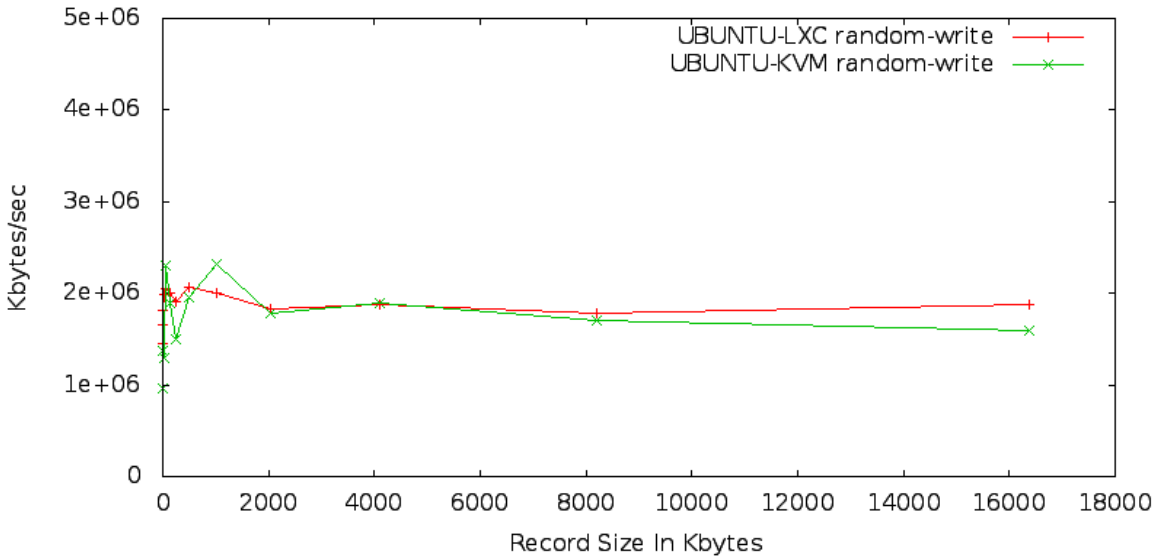

Fig. 3. Write Performance - LXC vs KVM

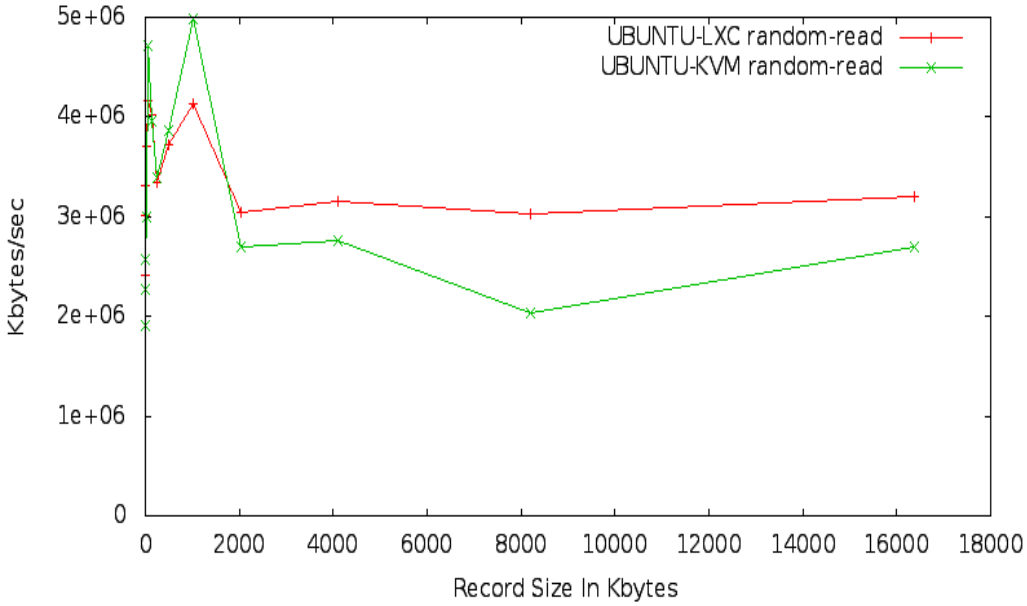

Fig. 4. Read Performance - LXC vs KVM

VI. CONCLUSIONS

Containers are gaining momentum as an alternative to virtual machines. They are light weight when compared to virtual machines. Both containers and hypervisors are existing technologies but used in a different forms. This paper elaborated on both these technologies and analyzed their performance in terms of I/O and computation. From the results obtained from the tests conducted Linux Containers showed better performance than the hypervisor (KVM). Since LXCs are lighter than hypervisor based VM, they provide a better alternative.

ACKNOWLEDGMENT

The authors thank R V College of Engineering and Rashtreeya Sikshana Samithi Trust for all the support. We also extend our thanks to the Director, Department of MCA, R V College of Engineering for the support and motivation.

REFERENCES

[1] Virtualixation : http://www.virtuatopia.com/index.php/An_Overview_of_Virtualization_and_VM wareServer2.0]

[2] Virtual Machines: Virtualization vs. Emulation : (http:// www. griffincaprio. com/ blog/ 2006/ 08/virtual-machines-virtualization-vs-emulation.html . Retrieved 2011-03-11.].

[3] Smith, James; Nair, Ravi (2005). "The Architecture of Virtual Machines". Computer (IEEE Computer Society) 38 (5): 32–38. doi:10.1109/MC.2005.173.

[4] Vmware : http://www.vmware. com/solutions/business- critical-app

[5] Turban, E; King, D; Lee, J; Viehland, D (2008). "Chapter 19: Building E-Commerce Applications and Infrastructure". Electronic Commerce A Managerial Perspective. pp. 27.

[6] "Virtualization in education" : http:/ / www-07. ibm. com/ solutions/ in/ education/ download/Virtualization in Education. pdf). IBM. October 2007.

[8] "Linux Containers" [online] https://wiki.archlinux.org/index.php/Linux_Containers

[9] "OpenVZ," 2012. [Online] http://www.openvz.org

[10] E.W. Biederman, "Multiple Instances of the Global Linux Namespaces," in Proceedings of the Linux, 2006.incomplete

[11] "Virtualization Technology," 2012. [Online]. Available: http://ark.intel.com/Products/Virtualization Technology

[12] "AMD Virtualization," 2012. [Online]. Available: http: //www.amd.com/br/products/technologies/ virtualization/

[13] N. Regola and J.-C. Ducom, "Recommendations for virtualization technologies in high performance computing," in Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, 30 2010-dec. 3 2010, pp. 409 –416.

[14] J. P. Walters, V. Chaudhary, M. Cha, S. G. Jr., and S. Gallo, "A comparison of virtualization technologies for hpc," Advanced Information Networking and Application

[15] S. Soltesz, H. P¨tzl , M. E. Fiuczynski, A. Bavier, and L. Peterson, "Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors," SIGOPS Oper. Syst. Rev., vol. 41, no. 3, pp. 275–287, Mar. 2007.

[16] "Isolation Benchmark Suite," 2012. [Online]. Available: http://web2.clarkson.edu/class/cs644/ isolation

