A COMPARISON OF LINUX CONTAINER AND VIRTUAL MACHINES

Prateek Jain
B.Tech 4th Year (CSE), Quantum School of Technology, Roorkee, Uttarakhand, India

Abstract
In the new era of Cloud computing world virtualization plays a vital role in isolating different resources which leads an organization to easily handling of its services however extra load of abstraction involved in virtualization reduces workload performance, which passed onto customer as worse experience. The advantage of container is that we can easily create copy of services as per the requirement while running that service. In this paper, we explore the performance evaluation of Linux containers and Virtual Machines. We compare them on the basis of launching time, memory and backup of system. We use virtual box as a type 2 Virtual Machine and Docker as a container manager. My result shows that the performance of container will be better or equal than the performance of VMs in almost all cases.
Keywords: Virtualization, Virtual Machines, Docker, Instance, Performance etc.

I. INTRODUCTION

In last few years Virtual Machines are used in large scale due to the Cloud computing. For providing services like Infrastructure-as-a-service(IaaS) vendors generally, use Virtual machines. Cloud platforms like Amazon, Azure, OpenStack make VMs available for its customers for running services like servers and databases. Many services like Platform as a service(PaaS), Software as a Service(SaaS), Network as a Service (NaaS) etc runs inside a VM with all their workload. The performance of VMs leads to the performance of overall Cloud services.

Container-based virtualization presents an interesting alternative to virtual machines in the cloud.[1] Virtual Private Server providers, which may be viewed as a precursor to cloud computing, have used containers for over a decade but many of them switched to VMs to provide more consistent performance. Although the concepts underlying containers such as namespaces are well understood[2], container technology languished until the desire for rapid deployment led PaaS providers to adopt and standardize it, leading to a renaissance in the use of containers to provide isolation and resource control. Linux is the preferred operating system for the cloud due to its zero price, large ecosystem, good hardware support, good performance, and reliability.

In this paper, I will analyze the performance of VM and containers by adding some workload like web servers, memory taken in first boot. Here I am not using type 1 Hypervisor like KVM, Microsofts Hyper-V, VMware ESX but I will use type 2 Hypervisor VM Oracle Virtual Box. I will not evaluate the case of containers running inside VMs or VMs running inside containers because we consider such double virtualization to be redundant (at least from a performance perspective). The fact that Linux can host both VMs and containers creates the opportunity for an apples-to-apples comparison between the two technologies with fewer confounding variables than many previous comparisons.

II. BACKGROUND

A. *Motivation and requirements for cloud virtualization*

Unix traditionally does not strongly implement the principle of least privilege, viz., “Every

program and every user of the system should operate using the least set of privileges necessary to complete the job." and the least common mechanism principle, viz., "Every shared mechanism ... represents a potential information path between users and must be designed with great care to be sure it does not unintentionally compromise security."[3]. Most objects in Unix, including the file system, processes, and the network stack are globally visible to all users. A problem caused by Unix's shared global file system is the lack
of configuration isolation. Multiple applications can have conflicting requirements for system-wide configuration settings. Shared library dependencies can be especially problematic since modern applications use many libraries and often different applications require different versions of the same library. When installing multiple applications on one operating system the cost of system administration can exceed the cost of the software itself. These weaknesses in common server operating systems have led administrators and developers to simplify deployment by installing each application on a separate OS copy, either on a dedicated server or in a virtual machine. Such isolation reverses the status quo compared to a shared server with explicit action required for sharing any code, data, or configuration between applications. Irrespective of the environment, customers want to get the performance they are paying for. Unlike enterprise consolidation scenarios where the infrastructure and workload are owned by the same company, in IaaS and PaaS there is an arms-length relationship between the provider and the customer. This makes it difficult to resolve performance anomalies, so PaaS providers usually provision fixed units of capacity (CPU cores and RAM) with no oversubscription. A virtualization system needs to enforce such resource isolation to be suitable for cloud infrastructure use.

B. Type 2 Virtual Machines

A Virtual Machine is a software virtualization package that installs on an operating system as an application. Virtual Machine allows additional operating systems to be installed on it, as a Guest OS, and run in a virtual environment. VirtualBox[4] is one of the most popular virtualization software application. Supported operating systems include Windows XP, Windows Vista, Windows 7, macOS X, Linux, Solaris, and OpenSolaris. These type of Virtual machines are mostly dependent on the size of RAM installed on the system.

C. Linux Container

Rather than running a full OS on virtual hardware, container-based virtualization modifies an existing OS to provide extra isolation. Generally, this involves adding a container ID to every process and adding new access control checks to every system call. Thus containers can be viewed as another level of access control in addition to the user and group permission system. In practice, Linux uses a more complex implementation described below. Linux containers are a concept built on the kernel namespaces feature, originally motivated by difficulties in dealing with high-performance computing clusters.[5] This feature, accessed by the clone() system call, allows creating separate instances of previously-global namespaces. Linux implements file system, PID, network, user, IPC, and hostname namespaces. For example, each file system namespace has its own root directory and mount table, similar to chroot() but more powerful.

III. EVALUATION

All the tests were performed on a Dell System Inspiron 3542 with two 1.7 GHz Intel core i5 processors. The system had 8 GB of RAM. I had used Red Hat Linux 7.3 64-bit with Linux Kernel 3.11.0, Docker 17.06, For consistency, all Docker containers used CentOS 7 base image and all VMs used RHEL 7.3 ISO image. RAM allotted to the Operating system running on the VM is 4 GB.

TABLE I. Comparison between Virtual Machine and Linux based Containers

	Virtual Machine	**Container (Docker)**
Time taken in launching a instance	>20 min.	< 2 sec.
RAM uses after boot	200 mb.	10 mb.
Backup of Instance	No	Yes
Backup Portability	No	Yes
Kernel	Own	Shared by base OS
GUI	Yes(Optional)	No

IV. CONCLUSION

In this paper, I had discussed about Virtual Machines and Linux based containers. We know that though Containers have some limitation but even after its good to use containers in place of Virtual Machines for some light weight applications.

Reference

[1].Stephen Soltesz, Herbert Potzl, Marc E. Fiuczynski, Andy Bavier, and ¨ Larry Peterson. Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors. In Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, EuroSys ’07, pages 275–287, 2007.

[2].Rob Pike, Dave Presotto, Ken Thompson, Howard Trickey, and Phil Winterbottom. The Use of Name Spaces in Plan 9. In Proceedings of the 5th Workshop on ACM SIGOPS European Workshop: Models and Paradigms for Distributed Systems Structuring, pages 1–5, 1992.

[3] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. In Proceedings of the IEEE, volume 63, Sep 1975.

[4].Ivan Sabolski, Hrvoje Leventić, Irena Galić Performance Evaluation of Virtualization Tools in Multi-Threaded Applications Volume 5, Number 2, 2014

[5]. E. W. Biederman. Multiple instances of the global Linux namespaces. In Proceedings of the 2006 Ottawa Linux Symposium, 2006.