LaTeX Linux Containers - Virtualization for Cloud Era

ASHOK VUPPADA[1]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: ashokmadhu66@gmail.com*

paper-1, March 27, 2017

In todays world , cloud is seen as delivery model for growing number of enterprises as the benefits of agility and efficiencies are better understood and available. The concept of virtualization is one of the building blocks for cloud infrastructure and services offering.The traditional VM work with hardware emulation causing inefficient resource and storage management. New technology 'Linux containers' build with the concept of 'Operating system Virtualization' aims at efficient resource management and customization.

Keywords: LXC,Cloud, I524

https://github.com/justbbusy/sp17-i524/tree/master/paper1/S17-IO-3024/report.pdf

1. INTRODUCTION

Traditional Hyperviser based Virtualization works with the idea of creating virtual hardware and running guest operating system on them. Applications are then expected to run on the guest operating system. The hypervisor layer would control the different guest operating systems.This model offers great deal of isolation,however the disadvantage of the this set up is each of the guest operating system would let a kernel process (memory and cpu manager processes) run on the host operating system and causing the performance overhead and resource constrain. It is also noted that each of the guest operating system runs from the ISO file (usually in GBs) causing limitations on the storage of the host. [1]

Container based virtualization works with the similar concept except eliminating the need of hypervisor layer. Each instance of the guest operating system can be assumed as any other processes running on the host system. There would be only one kernel processing running on the host operating system (that belong to the host OS) and it would shared with the guest operating systems. The hardware resources (CPU and memory) will also be shared. Since there is single process controlling the memory and CPU usage is running this model has performance advantage [1]

2. BUILDING BLOCKS

Linux containers are built on the modern kernel features like 'chroots' , 'namespaces' ,'cgroups' and Linux security models and mandatory access controls.[2]

- chroot : its a way of isolating a process to change only defined file system.The process running inside the chroot jail cannot act on the file system out side it. this is used to isolate the unstable applications to hamper the file system in use.[3]

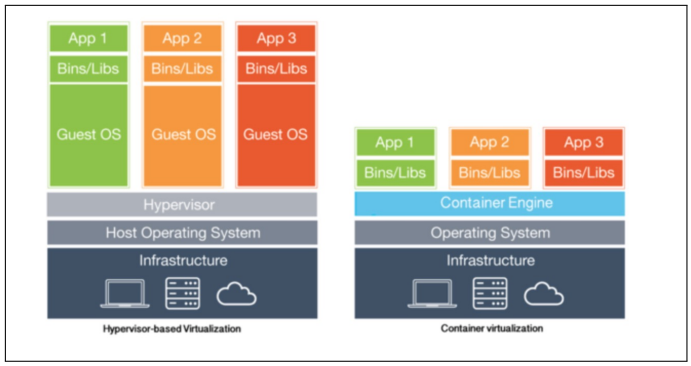

Fig. 1. Architecture Comparison of Hypervisor Based VM and containers.

- namespaces: namespaces provides the process level isolation for the global resources.It helps the process in each of the namespace to believe its the only process.[2]

- cgroups: its a kernel feature which allows to allocate resources to a given group of process(s).it helps one allocating and accounting CPU time, memory etc for a set of processes[4]

- Linux Security module: "Linux Security Modules (LSM) is a framework that allows the Linux kernel to support a variety of computer security models while avoiding favoritism toward any single security implementation".[5]

- Mandatory Access control: Its is process by which the security policies setup cannot gets overrides by a user. [6]

Linux containers are built on kernel features stated , as they are linux features the contender concept is not available in Microsoft windows.

3. PROJECT LXC AND DOCKER

Both LXC and Docker are the practical implementations of the linux containers. LXC is the original linux container developed.The project docker was initially started with the idea of extending LXC but ended up developing its own container. The key difference between LXC and a docker is in LXC each virtual container would allow multiple process to run where as in docker each process would need a separate docker container.Docker is designed to be stateless and credited for its portability. Because of the overwhelming popularity of docker it is often treated as synonym for linux containers .The below diagram depicts the main structural differences between LXC and docker.[7]

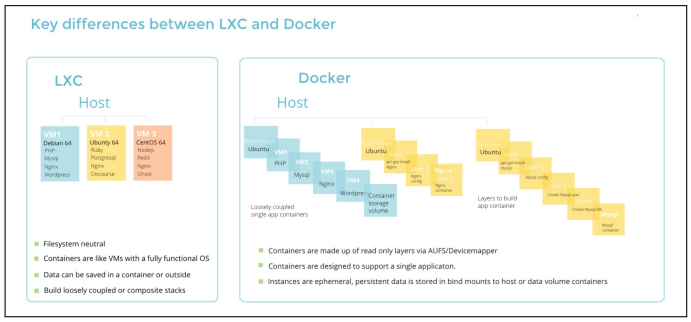

Fig. 2. Architecture Comparison of LXC and Docker.

4. PERFORMANCE COMPARISON

[8] In this study , Linux on KVM is studied for Hypervisor based VM , LXC and docker are compared as container based solutions. OSv is used as light weight guest OS.

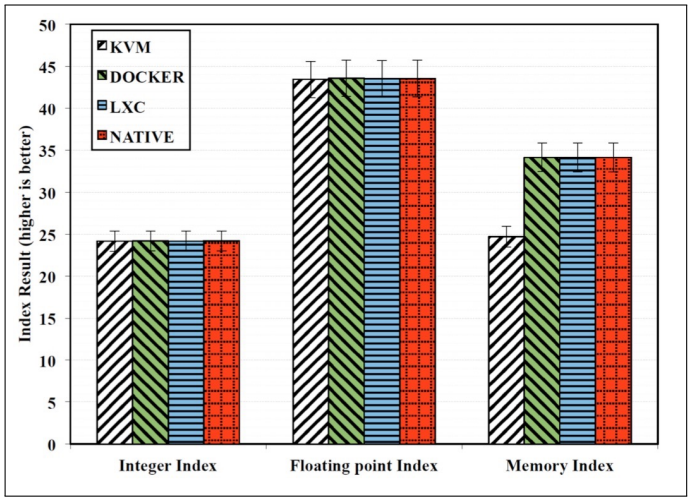

Fig. 3. Single Core Perf Comparison

1. CPU Performance
   CPU multithreaded performance is measured with a benchmark called 'Y Cruncher' which calculate the values of Pi the results can be seen from the table below NBENCH is used for single core comparison producing three different indexes: Integer Index, Floating Point Index, and Memory Index. Please see the blow diagram from the comparison

Platform	Multi-core Efficiency
Native	98.27%
LXC	98.19%
Docker	98.16%
KVM	97.51%

Fig. 4. Multi Core Perf Comparison

2. Memory Performance
   Below are test results using STREAM

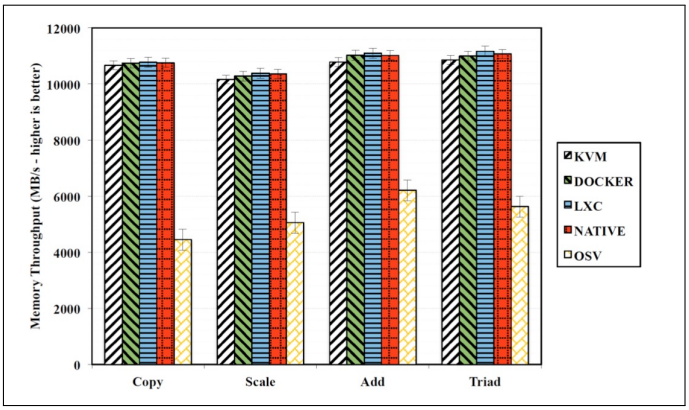

Fig. 5. Memory Perf Comparison

3. Disk I/O Performance
   Bonnie++ is used to measure the disk performance and below are the results.

Platform	Random Write speed (Kb/s)		Random Seeks	
Native	48254	%	1706	%
LXC	43172	**− 10.53%**	1517	**− 11.07%**
Docker	41170	**− 14.68%**	975	**− 42.84%**
KVM	23999	**− 50.26%**	125.7	**− 92.63%**

Fig. 6. Disk IO Perf Comparison

4. Network I/O Performance
   Network I/O is measured with Netperf

This test concludes that containers performed well but needs to improve on security and isolation.

Platform	TCP_STREAM (Mbps)		UDP_STREAM (Mbps)	
Native	9413.76	**%**	6907.98	**%**
LXC	9411.01	**− 0.00029%**	3996.89	**− 42.14%**
Docker	9412	**− 0.00018%**	3939.44	**− 42.97%**
KVM	6739.01	**− 28.41%**	3153.04	**− 54.35%**
OSv	6921.97	**− 26.46%**	3668.95	**− 46.88%**

Fig. 7. Network IO Comparison

5. USE CASES

[9]

1. Continous Integration
   With the concept of DevOps there is a tight integration between the development and Operations.The frequency of deploying new applications had increased tremondously.With multiple applications to be deployed with along with the dependencies,Op team used to have tough time, Container with the portability and packaging made the job easy. [9]

2. Container as a service
   This concept is similar to the PaaS, In a complex IT structure of an organization it was difficult to manage different existing technologies and add a new one,but with container as a service the new technology container image can be built and hosted on the existing host for the development to happen. This model eliminates the need of learning the dependencies associated with running multiple applications on the same host. [9]

3. Micro Services
   The idea of Micro service is to have the isolation between the application processes as much as possible so the each of the application can be built in the best technology suitable. This model is very helpful for modernization of the applications easily. Since container installation takes few seconds and doesn't come with the lot of dependencies they are the best choice for Microservices. [9]

6. LIMITATIONS

[10]

1. Containers are not suitable for all the purposes , some of the existing applications are designed best for sharing the physical layer of the machine.If scalability and fast deployment are not really applicable there is no good reason to use containers. [10]

2. Containers provides weaker isolation compare the hypervisor based virtual machines.Since they share the common kernel process, if there is some bug or virus on the host there would be chance to propgate to other containers. [10]

3. Containers are presently getting evolved,compared to the existing traditional VMs the tools available for monitoring the resources on containers are limited. [10]

4. containers are build on the kernel features build in Linux,Hence this cannot be directly used by other operating systems which does't have Linux kernel (Microsoft Windows)

7. CONCLUSION

In the era of cloud computing and micro services the need and popularity of containers are gradually growing up.Docker with its ability of "Build Ship Run" [11] ability made the container concept more popular and useful.But eventually its not going to replace the traditional hypervisor based virtual machines any time soon (as they are far more superior interns of security and integration compared to containers). Both the technologies needs to be used in conjunction to get the better of both the worlds.

REFERENCES

[1] "Hypervisor and container based virtulization," web page. [Online]. Available: http://www.slashroot.in/difference-between-hypervisor-virtualization-and-container-virtualization
[2] "Linux container building blocks," web page. [Online]. Available: http://bodenr.blogspot.com/2014/03/linux-containers-building-blocks.html
[3] "Basic chroot," web page. [Online]. Available: https://help.ubuntu.com/community/BasicChroot
[4] "Lxc and docker explained," web page. [Online]. Available: http://www.infoworld.com/article/3072929/linux/containers-101-linux-containers-and-docker-explained.html
[5] "Linux security modules," web page. [Online]. Available: https://en.wikipedia.org/wiki/Linux_Security_Modules
[6] "Linux madandatory access controls," web page. [Online]. Available: https://www.linux.com/news/securing-linux-mandatory-access-controls
[7] "Understanding the key differences between lxc and docker," web page. [Online]. Available: https://www.flockport.com/lxc-vs-docker/
[8] M. K. Roberto Morabito, Jimmy Kjällman, "Hypervisors vs. lightweight virtualization: a performance comparison," web page, 2015. [Online]. Available: https://pdfs.semanticscholar.org/551d/2b55067c06b5667cfc35b3b20096b9235cb4.pdf
[9] "Use cases for conatiners," web page. [Online]. Available: https://www.virtualizationpractice.com/container-use-cases-35048/
[10] "Cons for container technology," web page. [Online]. Available: http://searchservervirtualization.techtarget.com/feature/Five-cons-of-container-technology
[11] "Docker," web page. [Online]. Available: https://www.docker.com