IJCSE International Journal of Computer Sciences and Engineering Open Access
Survey Paper Volume-4, Issue-7 E-ISSN: 2347-2693

# A Survey of Performance Comparison between Virtual Machines and Containers

Prashant Ramchandra Desai

*Department of Computer Engineering, Bharati Vidyapeeth University, Pune, India*

**Available online at: www.ijcseonline.org**

Received:18/Jun/2016 Revised: 26/Jun/2016 Accepted: 16/Jul/2016 Published: 31/Jul/2016

***Abstract*—**: Since the onset of Cloud computing and its inroads into infrastructure as a service, Virtualization has become peak of importance in the field of abstraction and resource management. However, these additional layers of abstraction provided by virtualization come at a trade-off between performance and cost in a cloud environment where everything is on a pay-per-use basis. Containers which are perceived to be the future of virtualization are developed to address these issues. This study paper scrutinizes the performance of a conventional virtual machine and contrasts them with the containers. We cover the critical assessment of each parameter and its behavior when its subjected to various stress tests. We discuss the implementations and their performance metrics to help us draw conclusions on which one is ideal to use for desired needs. After assessment of the result and discussion of the limitations, we conclude with prospects for future research.

***Keywords*—***Performance, Virtual Machines(VMs), Containers, Virtualization, Kernel Virtual machines (KVM), Docker, Hypervisor.*

## I. INTRODUCTION

The abundance of cloud computing over the past decade has lead to significant growth in virtualization techniques used. The services provided by Cloud Computing indirectly depend on the level of virtualization provided by the underlying virtual machines. Hardware virtualization consists of abstracting complete hardware resources so that entire software like an operating system can run on it. Hardware virtualization is categorized further depending upon the levels of abstraction it offers into full virtualization, partial virtualization, and Para virtualization. In Para virtualization, the guest operating system is recompiled before being installed in a virtual machine. The hypervisor serves as a host for the guest OS and adds a layer of virtualization. It also acts as an interface between guest OS and the hardware. In full virtualization, the hypervisor is directly installed on the hardware. Each guest operating gets all the features of the underlying hardware because of the layer of abstraction provided it doesn't feel the hypervisor's presence. It is a commonly implemented form of virtualization in virtual machines [1][2]. Since the virtualization software is a bundle of hardware drivers and interfaces which needs to be pre-installed before creating the virtual machines. It is been observed, in some cases [1]some of the interfaces and drivers do not function as expected thus enabling the usage of Dockers where a small size Unix/Linux operating system is installed on the bare-metal and the virtual machines are hosted from the Docker or the container-based virtualization. Container-based virtualization is an alternate technology to virtual machines and is quickly replacing them in the cloud environment. The virtualization layer runs as an application, where the operating system's kernel runs on bare metal with guest OS installed on top of it. These are known as containers.

## II. BACKGROUND

### A. KVM

Kernel-based Virtual Machine (KVM) is a virtualization technology for Linux systems that turns it into a hypervisor and is designed for x86 processor architecture. The hypervisor is built on Linux kernel, running an open source operating system. It provides support for multiple guest OS include windows, BSD, and Solaris. KVM is a full virtualization technique where you can provide virtualized hardware: that allocates the number of CPUs, memory, hard disk space to the guest operating system.

There are a number of management tools as libvirt, oVirt and Virtual Machine Manager available to easily manage KVM through graphical user interface . One of the features supported by KVM is the live migration in which entire data centers can be backed up for maintenance without affecting the functioning of the guest operating system.KVM supports "hot plugging" that means you can resize or add additional resources to the virtual machine while it is in operation without disrupting its services.

### B. Containers

Containers are similar to virtual machines in the services they provide except they don't come with an overhead of running a separate kernel and virtualizing all hardware components. It modifies the host operating system to provide abstraction. They have a container ID and group permission system. Its main concept is built on the use of kernel

 

namespaces where isolated containers are created. These have no access to objects outside the container. A container which runs a full operating system is a system container whereas the one which runs an application is an application container. Because a containerized system has only one kernel, there is only one level of resource scheduling and allocation. The process that run inside a container is not aware of the limit on the resources it can utilize hence there always remains a risk where an application may over-allocate itself resources.[3]

Security in containers is achieved by managing the group permissions and by creating namespace awareness, where the users are not treated with the same privileges inside the container as those outside them. Docker is the most widely used container today that deploys applications inside software containers along with its code, runtime, system tools, system libraries, etc . Docker has layered file system images, supported by AUFS (Another UnionFS). Containers have a faster boot time than virtual machines.

## III. ASSESSMENT

A virtual machine can be benchmarked using various performance measurements tools that integrate with your hypervisor accordingly. Some of the tools used are VMware ESXi, VSphere, vCenter. These tools monitor the resource utilization of an active virtual machine and later aggregate data into excel format for analysis. These same tools can be used to benchmark application or system containers.

The performance can be measured on the basis of various parameters such as throughput, latency, bandwidth, etc of the system. We are mainly concerned with the overhead generated by native, virtualized and non-virtualized environments and how it can be optimized to improve system performance and prove which one is suitable to use under preferred conditions. [7]-[10]

The behavior of various parameters under the defined workload is observed carefully and recorded, to study its characteristics and get a consolidated comparison between the three implementations.

### *A. CPU Performance*

Throughput is the parameter used to measure the output of the workload the CPU is subjected to a compression, High Performance Computing (HPC) test. As observed the native and Docker performance is similar in compression while KVM is slower as compared to them.

HPC performance is similar on native and Docker but is reasonably slow on KVM because of abstraction that works as a disadvantage in this situation. The CPU schedulers do not influence the processor in the native and Docker setup hence there is no difference in performance.

### *B. Memory Performance*

Bandwidth is the parameter used to measure the speed of memory access and operations. Under various benchmarks developed to stress test the memory in sequential and random access methods, it is observed that the performance of native, Docker and KVM environment is almost the same for various operations with very little deviation. The testing was carried out on a single node on large datasets. Container based systems have the ability to return unused memory to the host resulting in better usage. Para virtualization systems suffer from double cache since the same memory blocks are used by the host and the virtual machine.

### *C. Network Performance*

Bandwidth is used to measure the performance of the network communication. The communication scenario is bulk data transfer over a single TCP connection like the client -server model.

The data transfer rate is measured in both directions since TCP/IP stack has different policies for sending and receiving data. The major component that causes a bottleneck in performance is the NIC in which case we use the CPU cycles to measure the overhead. As for the performance goes Docker uses bridging and NAT which increases the route length. Dockers which do not use NAT have same performance to native systems.KVM performance can be improved if the VM can directly communicate with the host bypassing the in-between layers.

Latency can also be another network parameter used to determine performance. In the above setup of the experiment, the client sends 200 bytes of request to the server and the server response with a 400 bytes reply in this case the client has to wait to process the complete request. If we consider wireless and wired communication protocols Docker with NAT doubles latency time KVM also adds to latency time compared to its native system implementations. The difference in network performance is due to the virtualized network devices implemented by virtual machines and Docker.

### *D. Disk Performance*

Throughput is used to measure the efficiency of the disk operations. As duly noted for sequential read and sequential write operations, the Docker and KVM add very little overhead when compared to native, but there is a lot of performance variance in KVM's case cause of suspected bottleneck in fiber channel.

For random read and random write operations, Docker has no overhead, but KVM's performance decreases

 

significantly. Disk performance is affected by the I/O scheduler used by the system.

*E. Application Performance-Redis*

Redis (REmote Dictionary server) is the most popular NoSQL database used in containers and is open source. It is an in-memory data store and key-valued database.

Generalized operations between client and server where performance issues are noticed only in the form of network latency. This arises due to the large number of concurrent requests the client makes simultaneously to the server. Redis is widely used in the cloud environment. In this test scenario, the number of requests made simultaneously is ten times the number of clients. Redis is extensively used to test the networking and memory subsystem.

The native environment can handle a large number of clients connected to the networking subsystem, so we can easily scale the number of clients connected. The bottleneck in performance is visible at the CPU because Redis is a single-threaded application; it increases the latency of the CPU. Docker's performance is virtually similar to that of native except with NAT enabled, latency increases as new connections are added. KVM's performance is lower initially but approaches that of native systems as concurrency increases. Hence we can draw the conclusion that Redis application needs to be concurrent to be fully utilized.

*F. Application Performance-MySQL*

MySQL is a relational database that can be used to stress the memory, file system, networking and inter-process communication subsystems. Various open source benchmarks can be run against the database to test the transaction throughput of the system. Various configurations were used to test MySQL, running under the native system, MySQL under Docker that uses the host's networking, MySQL under Docker using NAT for networking and KVM running MySQL.

Throughput is the number of transactions per second increase until the peak point where it becomes stable. Docker has similar performance to that of native systems; KVM has higher overhead compared to all other configurations used to demonstrate. The Docker's layered file system also introduces overhead because of the I/O request getting redirected through various layers. The latency of the system increases with the load, but Docker does not show this observation when compared to other to other setups because of lower throughput at lower concurrency rates. When it comes to CPU peak utilization native system is able to achieve a higher rate as compared to Dockers which show that Dockers have a small but significant impact. [4]-[16]

CPU utilization measured in throughput is minimal for the same amount of CPU used for Docker and Docker with NAT, but the latency for Docker is higher for lower values of concurrency. This is due to mutex contention which prevents MySQL from fully utilizing the CPU for configurations. [17]-[26]

*G. Discussion*

We can analyze the results, as expected for the given environment containers and VMs generate a negligible amount of overhead when it comes to CPU and memory usage. When it comes to network usage both Docker and KVM add to the latency and hence need to tune and are of no good with their standard implementations. Most of the overhead added are because of the bottlenecks in end communication devices. When it comes to disk performance Docker and native are similar to each other but KVM sees a considerable downfall in performance since each operation has to pass through AUFS.

The major limitation of virtualization technology is bottlenecks in I/O intensive applications. The idea is to get very close to native systems when it comes to I/O processing and decrease as many layers as possible in virtual machines and containers. Container-based virtualization system has better CPU and I/O performance because of its ability to release used unused and resources and work in isolation.

## CONCLUSIONS AND FUTURE WORK

This study analyzes the behavior of various well-known parameters, under different workload conditions to give us a comprehensive performance comparison between Containers and Virtual Machines. As the survey clearly states, that no one, technology can be substituted by another right away, each having its own merits and demerits and each technology can be used efficiently with proper configuration suiting your particular needs.

Some of the areas which are not researched to its full extent are:

- Performance analysis in isolation when more than more than one workload is executing on the server.
- The performance of different kinds of workloads such as data intensive workloads which are more I/O bound in nature.
- Performance trade-off between live migration and restarting virtual machines and containers.

## REFERENCES

[1] Xavier, M.G., Neves, M.V. and De Rose, C.A.F., 2014, February. A performance comparison of container-based virtualization systems for mapreduce clusters. In 2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (pp. 299-306). IEEE.

 

[2] Soltesz, S., Pötzl, H., Fiuczynski, M.E., Bavier, A. and Peterson, L., 2007, March. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. In ACM SIGOPS Operating Systems Review (Vol. 41, No. 3, pp. 275-287). ACM.

[3] Vaughan-Nichols, S.J., 2006. New approach to virtualization is a lightweight.Computer, 39(11), pp.12-14.

[4] Raval, K.S., Suryawanshi, R.S., Naveenkumar, J. and Thakore, D.M., 2011. The Anatomy of a Small-Scale Document Search Engine Tool: Incorporating a new Ranking Algorithm. International Journal of Engineering Science and Technology, 1(3), pp.5802-5808.

[5] Archana, R.C., Naveenkumar, J. and Patil, S.H., 2011. Iris Image Pre-Processing And Minutiae Points Extraction. International Journal of Computer Science and Information Security, 9(6), p.171.

[6] Jayakumar, M.N., Zaeimfar, M.F., Joshi, M.M. and Joshi, S.D., 2014. INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING & TECHNOLOGY (IJCET). Journal Impact Factor, 5(1), pp.46-51.

[7] Naveenkumar, J. and Joshi, S.D., 2015. Evaluation of Active Storage System Realized through MobilityRPC.

[8] Jayakumar, D.T. and Naveenkumar, R., 2012. SDjoshi,". International Journal of Advanced Research in Computer Science and Software Engineering," Int. J, 2(9), pp.62-70.

[9] Jayakumar, N., Singh, S., Patil, S.H. and Joshi, S.D., Evaluation Parameters of Infrastructure Resources Required for Integrating Parallel Computing Algorithm and Distributed File System.

[10] Jayakumar, N., Bhardwaj, T., Pant, K., Joshi, S.D. and Patil, S.H., A Holistic Approach for Performance Analysis of Embedded Storage Array.

[11] Naveenkumar, J., Makwana, R., Joshi, S.D. and Thakore, D.M., 2015. OFFLOADING COMPRESSION AND DECOMPRESSION LOGIC CLOSER TO VIDEO FILES USING REMOTE PROCEDURE CALL. Journal Impact Factor, 6(3), pp.37-45.

[12] Naveenkumar, J., Makwana, R., Joshi, S.D. and Thakore, D.M., 2015. Performance Impact Analysis of Application Implemented on Active Storage Framework. International Journal, 5(2).

[13] Salunkhe, R., Kadam, A.D., Jayakumar, N. and Thakore, D., In Search of a Scalable File System State-of-the-art File Systems Review and Map view of new Scalable File system.

[14] Salunkhe, R., Kadam, A.D., Jayakumar, N. and Joshi, S., Luster A Scalable Architecture File System: A Research Implementation on Active Storage Array Framework with Luster file System.

[15] Jayakumar, N., Reducts and Discretization Concepts, tools for Predicting Student's Performance.

[16] Jayakumar, M.N., Zaeimfar, M.F., Joshi, M.M. and Joshi, S.D., 2014. INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING & TECHNOLOGY (IJCET). Journal Impact Factor, 5(1), pp.46-51.

[17] Kumar, N., Angral, S. and Sharma, R., 2014. Integrating Intrusion Detection System with Network Monitoring. International Journal of Scientific and Research Publications, 4, pp.1-4.

[18] Namdeo, J. and Jayakumar, N., 2014. Predicting Students Performance Using Data Mining Technique with Rough Set Theory Concepts. International Journal, 2(2).

[19] Naveenkumar, J., Keyword Extraction through Applying Rules of Association and Threshold Values. International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), ISSN, pp.2278-1021.

[20] Kakamanshadi, G., Naveenkumar, J. and Patil, S.H., 2011. A Method to Find Shortest Reliable Path by Hardware Testing and Software Implementation. International Journal of Engineering Science and Technology (IJEST), ISSN, pp.0975-5462.

[21] Naveenkumar, J. and Raval, K.S., Clouds Explained Using Use-Case Scenarios.

[22] Naveenkumar J, S.D.J., 2015. Evaluation of Active Storage System Realized Through Hadoop. International Journal of Computer Science and Mobile Computing, 4(12), pp.67–73.

[23] Rishikesh Salunkhe, N.J., 2016. Query Bound Application Offloading: Approach Towards Increase Performance of Big Data Computing. Journal of Emerging Technologies and Innovative Research, 3(6), pp.188–191.

[24] Sagar S lad s d joshi, N.J., 2015. Comparison study on Hadoop's HDFS with Lustre File System. International Journal of Scientific Engineering and Applied Science, 1(8), pp.491–494.

[25] Salunkhe, R. et al., 2015. In Search of a Scalable File System State-of-the-art File Systems Review and Map view of new Scalable File system. International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) - 2016. pp. 1–8.

[26] Naveenkumar, J. and Raval, K.S., Clouds Explained Using Use-Case Scenarios. INDIACom-2011 Computing for Nation Development, 2011/3.

[27] N. Jayakumar, "Reducts and Discretization Concepts, tools for Predicting Student's Performance," Int. J. Eng. Sci. Innov. Technol., vol. 3, no. 2, pp. 7–15, 2014.

[28] BVDUCOE, B.B., 2011. Iris Image Pre-Processing and Minutiae Points Extraction. International Journal of Computer Science & Information Security.

[29] P. D. S. D. J. Naveenkumar J, "Evaluation of Active Storage System Realized through MobilityRPC," Int. J. Innov. Res. Comput. Commun. Eng., vol. 3, no. 11, pp. 11329–11335, 2015.

 

[30] Anjali Nigam and Vineet Singh, "Securing Data Transmission in Cloud using Encryption Algorithms", International Journal of Computer Sciences and Engineering Research Volume 4, Issue 6, June 2016.
[31] Pritika Goel, "An Improved Load Balancing Technique in Weighted Clustering Algorithm", International Journal of Computer Sciences and Engineering Research Volume 4, Issue 6, June 2016.
[32] Anish Babu S , Hareesh M J , John Paul Martin , Sijo Cherian and Yedhu Sastri."System Performance evaluation of Paravirtualization, Container virtualization and Full virtualization using Xen, OpenVZ and XenServer" Fourth International Conference on Advances in Computing and Communications.pp 247-250 IEEE 2014

## AUTHORS PROFILE

**Prashant Ramchandra Desai** has received his B.E(2014) in computer engineering from Karmeer Bhaurao Patil College of Engineering , Satara. He is currently pursuing M.Tech in computer engineering from Bharati Vidyapeeth College Of Engineering ,Pune. His area of interest include virtualization,Big Data and storage area network.

 