Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 171 (2020) 1419–1428

Procedia Computer Science

www.elsevier.com/locate/procedia

Third International Conference on Computing and Network Communications (CoCoNet'19)

Performance Evaluation of Docker Container and Virtual Machine

Amit M Potdar[a], Narayan D G[b], Shivaraj Kengond[c], Mohammed Moin Mulla[d]

[a.b.c.d]*School of Computer Science and Engineering, KLE Technological University, Hubballi,580031, India*

Abstract

Server virtualization is a technological innovation broadly used in IT enterprises. Virtualization provides a platform to run different services of operating systems on the cloud. It facilitates to build multiple virtual machines on a single basic physical machine either in the form of hypervisors or containers. To host many microservice applications, the emergent technology has introduced a model which consists of different operations performed by smaller individual deployed services. Thus, the demand for low-overhead virtualization technique is rapidly developing. There are many lightweight virtualization technologies; docker is one among them, which is an open-source platform. This technology allows developers and system admins to build, create, and run applications using docker engine. This paper provides the performance evaluation of Docker containers and virtual machines using standard benchmark tools such as Sysbench, Phoronix, and Apache benchmark, which include CPU performance, Memory throughput, Storage read/write performance, load test, and operation speed measurement.

© 2020 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)
Peer-review under responsibility of the scientific committee of the Third International Conference on Computing and Network Communications (CoCoNet'19).

Keywords: Virtualization; Docker Container; Virtual Machine; Benchmark tools.

1. Introduction

In recent years, attention to Cloud computing is increasing. Numerous technologies are developed by the IT Industries on the emergence of Xen, HyperV, VMware vSphere, KVM, etc., which are known as virtualization technologies. To deploy many applications on the same virtual machine, applications and dependencies need to be organized and isolated. Because of virtualization, multiple applications can run on the same physical hardware. Drawbacks of virtualization techniques are: virtual machines are large in size, an unstable performance due to running multiple virtual machines, the boot-up process takes a long time to run, and virtual machines are unable to solve difficulties like manageability, software updates, and continuous integration/delivery. These problems led to

* Corresponding author.
E-mail address: amitpotdar31@gmail.com

1877-0509 © 2020 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)
Peer-review under responsibility of the scientific committee of the Third International Conference on Computing and Network Communications (CoCoNet'19).
10.1016/j.procs.2020.04.152

the emergence of a new process called Containerization that further led to virtualization at the OS level, whereas virtualization brings absorption to the hardware level. Containerization uses hosts operating system which shares relevant libraries and resources. It is more efficient because there is an absence of a guest OS. On the host kernel, the application-specific binaries and libraries can be processed, in which it makes execution very fast. The containers are formed with the help of the Docker platform, which combines applications and their dependencies. These containers are always being executed on top of the operating system's kernel in isolated space. This containerization feature of Docker makes sure that the environment supports any related application [1].

In this work, to quantify and contrast applications over hypervisor-based virtual machine and Docker container, a series of experiments are carried out. These tests help us to understand the performance implications of two major virtualization technologies - containers and hypervisors. The paper is organized as follows; section 2 gives the background study and a brief explanation about technologies and platforms. Section 3 addresses the methodology used to understand performance comparison. In Section 4, benchmarking results are presented. Finally, in Section 5, conclusion and future work are provided.

2. Background Study

2.1 Docker

Containerization is a technology that combines the application, related dependencies, and system libraries organized to build in the form of a container. The applications which are built and organized can be executed and deployed as a container. This platform is known as Docker, which makes sure that application works in every environment. It also automates the applications that will deploy into Containers. The Docker appends an additional layer of deployment engine over a container environment where the applications are executed and virtualized. To run a code efficiently, Docker helps to provide a quick and lightweight environment. The four main parts of Docker: Docker Containers, Docker Client-Server, Docker Images, and Docker Engine. The following sections will have a detailed explanation of these components

2.1.1 Docker Engine

The essential part of the Docker system is Docker Engine, a Client-server application which is installed on the host machine with the following components.

(a) Docker Daemon: a type of long-running program (the dockerd command) which helps to create, build, run application.
(b) A Rest API is used to communicate to the docker daemon.
(c) A client sends a request to docker daemon through the terminal to access the operations.

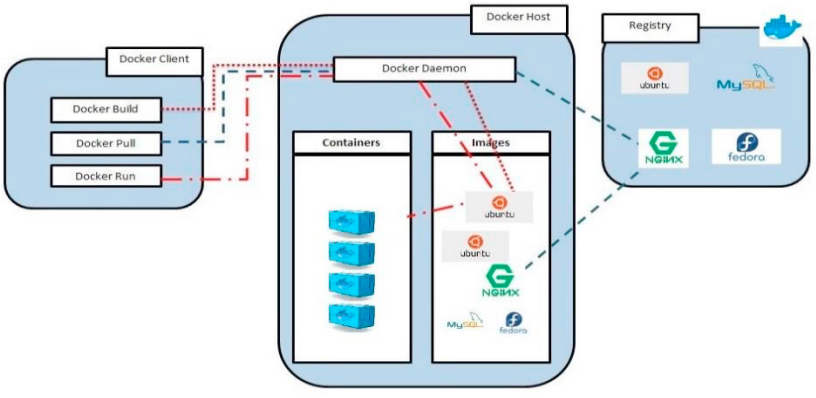

Fig. 1. Architecture of Docker Container

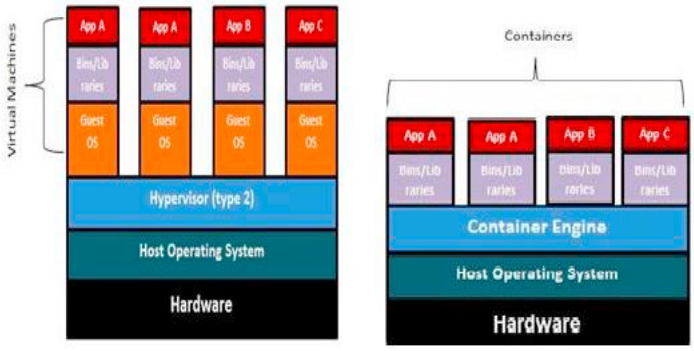

Fig. 2. (a) Hypervisor based architecture (b) Container based architecture

2.1.2 Docker Client-Server

Docker technology mainly refers to client-server architecture. The client communicates to the Docker daemon, which acts as a server that is present within the host machine. The daemon works as three major processes running, building, and distributing containers. Both the Docker container and daemon can be placed in a single machine. Fig. 1. shows the Docker architecture.

2.1.3 Docker Images

Docker Images can be built by two methods. The primary method is to build an image with the help of a read-only template. The template consists of base images, either it can be Operating System such as centos, ubuntu 16.04 or fedora, or any other base OS images that are lightweight. Generally, base images are the foundation of every image. It is necessary to build a new image whenever base images are created from scratch. This type of creating a new image is called “committing a change.” The next method is to create a docker file which has all the instructions for creating a docker image. When the docker build command is executed from the terminal, the image will be created with all the dependencies which are mentioned in the docker file. This process is known as an automated method of building an image.

2.1.4 Docker Containers

Docker Containers are created by Docker image. To run the application in a confined way, every kit required for the application is to be held by the container. The container images can be created based on the service requirement for the application or software. Suppose an application that includes Ubuntu OS and Nginx server should be appended into the docker file. Using the command “docker run,” container with the image of Ubuntu OS consisting of the Nginx server is created and starts running.

2.1.5 Comparison between Virtual Machine and Docker Container

Docker is sometimes referred to as lightweight VMs, but they are not VMs. The underlying technology, as discussed in the below Table 1, differences in virtualization technologies. Fig. 2. shows the architecture of the virtual machine and docker container.

Table 1. Virtual Machine versus Docker Container

	Virtual Machines	Docker Containers
Isolation Process Level	Hardware	Operating System
Operating System	Separated	Shared
Boot up time	Long	Short
Resources usage	More	Less
Pre-built Images	Hard to find and manage	Already available for home server
Customised preconfigured images	Hard to build	Easy to build
Size	Bigger because they contain whole OS underneath	Smaller with only docker engines over the host OS
Mobility	Easy to move to a new host OS	Destroyed and recreated instead of moving.
Creation time	Longer	Within seconds

2.2 Related Work

The use of developing virtual machines (VMs) is common in organizations. VMs are widely to execute complex tasks such as Hadoop [2]. However, users also use VMs even for launching a small application, which makes the system inefficient. There is a need for launching a lightweight application, which is faster and makes the system efficient. Docker container is one of the technologies offering lightweight virtualization, and this motivates us to carry out the background work. Fig 1 depicts the architecture of Docker Container.

In [3], authors present an overview of the performance evaluation of virtual machine and Docker containers in terms of CPU performance, Memory throughput, Disk I/O, and operation speed measurement. The authors in [4] focused on the implementation of Docker containers in the HPC Cluster. In the later part of the paper, the authors explain about different implementation approaches for choosing the container model and usage of LNPACK and BLAS. In [5], authors discuss lightweight virtualization approaches, in which addressing of containers and unikernel issues. Further, the paper also discusses the statistical evaluation of ANOVA test and a post-hoc comparison using the Tukey method of the collected data. The author also discusses the different benchmarking tools used for the comparison of unikernel and containers. Static HTTP Server and key-value store parameters are used for the experimental analysis for application performance, which is deployed over the cloud. Nginx server is used for the HTTP performance, and to measure get and set operations, Redis benchmark is used.

The authors in [6] discuss the evaluation with benchmarking applications using KVM, Docker, and OSv. In [7], the authors discuss a brief survey on Virtual Machines and Containerized Technologies. It also discusses docker and docker performance with various parameters CPU, Memory throughput, Disk I/O. In [8] the author discusses the fundamental concepts of docker architecture, components of docker, docker images, docker registries, docker client, and server architecture. Difference between virtual machines and docker container are discussed. In [9], authors explain about performance comparison of container-based technologies for the cloud with a different set of parameters. The paper provides information about the usage of OpenStack based cloud deployments, which are considered for comparison. The platforms which are used for the performance comparison are docker, LXC, and flockport. The authors in [10] discuss performance comparison of virtual machine and docker with respect to various parameters such as CPU, network, disk, and two real server applications, which are redis and MySql.

3. Methodology

In this section, the evaluation of KVM and Docker are performed using benchmarking tools. The following benchmarking tools used for the performance evaluation are Sysbench [11], Phoronix [12], Apache benchmark [13]. These benchmark tools measure CPU performance, Memory throughput, Storage read and writes, Load test, and operation speed measurement. The two HP servers are used for performance evaluation concerning various

parameters, for this one server is used as a virtual machine which is installed on top of Host OS (window 10) and whereas guest OS (Ubuntu 16.04) with addition to this docker engine is installed on top of the virtual machine and another server as a bare metal with host OS Ubuntu 16.04 and docker engine is installed on it. All the tests were performed on an HP server with two Intel Xenon E5-2620 v3 processors at 2.40 GHz at a total of 12 cores and 64 GB RAM. Ubuntu 16.04 64-bit with Linux kernel 3.10.0 was used to perform all the tests. To maintain consistency and uniformity, the same operating system, Ubuntu 16.04, was used as the base image for all Docker containers. The 12vCPUs and sufficient RAM are configured for VM. Fig. 3. presents the evaluation methodology of different virtualization technologies with various benchmarking tools.

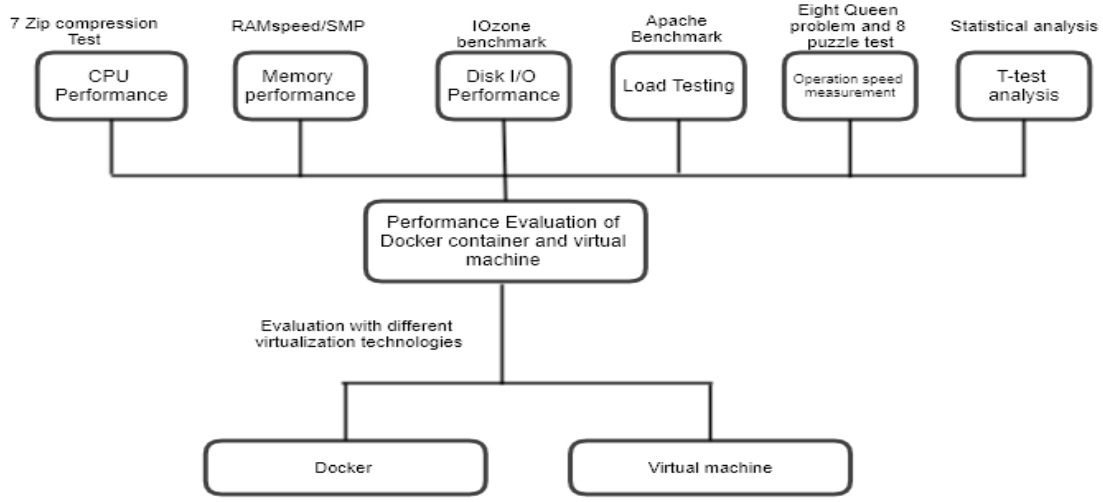

Fig. 3. Performance evaluation with various benchmarking tools

4. Results and Discussions

This section discusses the performance analysis of virtualization technologies. Results are classified into four subsections. Section 4.1 describes all the CPU measurements, memory throughput; Storage read, and writes measurements are shown in sections 4.2 and 4.3. Section 4.4 presents a load test analysis. Section 4.5 describes operation speed measurement, which consists of two tests: Eight Queen problem and eight puzzle problems. Finally, in section 4.6, a statistical t-test analysis is carried out.

4.1 CPU Performance

Computing performance can be measured by the number of operations the system has performed within a given time (events/sec) or by the completion time of a certain task. The results mainly depend on the number of virtual CPU cores that are allocated to the server. Testing CPU performance comparison has been done through the following tools sysbench, Phoronix, and Apache benchmark.

4.1.1 Maximum Prime Number Operation

On the Sysbench tool test conducted to find out the time required to perform the maximum prime number. The maximum prime number value for the operation is taken to be 50000 with time of 60 seconds and 4 thread operations. From Fig. 4. it is observed that docker container takes much lesser time to execute the operation when compared with VM. This is due to the hypervisor presences in the virtual machine. Therefore, it takes more time for the execution.

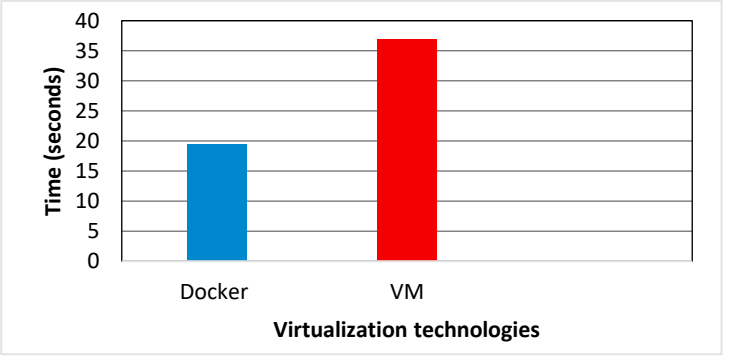

Fig. 4. CPU Comparison of Docker and Virtual Machine

4.1.2 7 Zip Compression Test

7-Zip is an open-source file archiver, which is service for compression of a group of files into containers known as "archives." The two tests for LZMA benchmark: Compressing and Decompressing LZMA methods. This test measures the time needed to compress a file using 7 Zip compressions. The file size used for the compression test is 10GB. Fig 5 shows the CPU performance comparison with 7 Zip compression test. Hence according to results obtained, Docker Container performance is much better than VM when performed with a large amount of file compression.

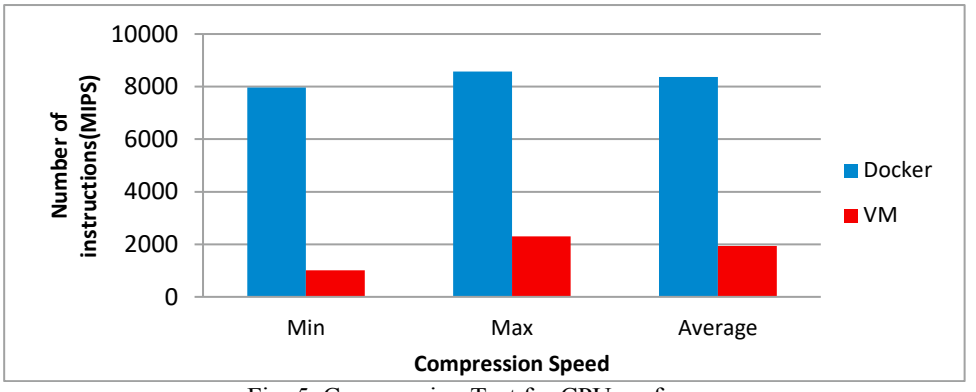

Fig. 5. Compression Test for CPU performance

4.2 Memory Performance

RAM speed/SMP (Symmetric Multiprocessing) is a cache and memory benchmarking tool used for the measurement of RAM speed for virtualization technologies, i.e., docker and Virtual Machine. Fig. 6. represents RAM Speed comparisons between virtualization technologies. The following two major parameters are considered while testing the RAM speed. INTmark and FLOATmark components are used in the RAM Speed SMP benchmarking tool, which measures the maximum possible cache and memory performance while reading and writing individual blocks of data.

INTmem and FLOATmem, they are synthetic simulations but balanced closely with the real world of computing. Each consists of four subtests (Copy, Scale, Add, Triad) to measure different aspects of memory performance. The transfer of data from one memory location to another is done by copy command, i.e. (X = Y). The modification of data before writing is multiplied with a certain constant value is done by scale command, i.e. (X = n*Y). The data is read from the first memory location and then from second when commands ADD is called. Then the resultant data is placed in third place (X = Y + Z). Triad is a combination of Add and Scale. The data is read

from the first memory location to scale and then add from the second place to write it to the third place (X = n*Y + Z). Fig. 6. shows memory performance with respect to the RAM Speed SMP test.

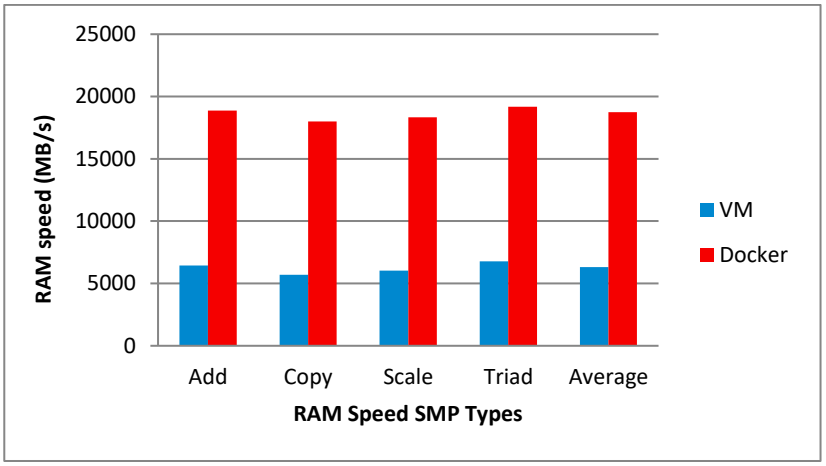

Fig. 6. RAM Speed comparisons between virtualization technologies

4.3 Disk I/O Performance

To test the hard disk performance, the IOzone benchmark tool is used for the performance analysis. For testing the operations such as write and read of the system, a record size 1MB and file size of 4GB were used. It can infer from Fig 7 that the performance of Docker is much better when compared with the virtual machine. The disk writes and reads operations of a VM are reduced by more than half of that of Docker container (approximately 54%). Fig. 7. shows the Disk performance of both virtual machines and Docker.

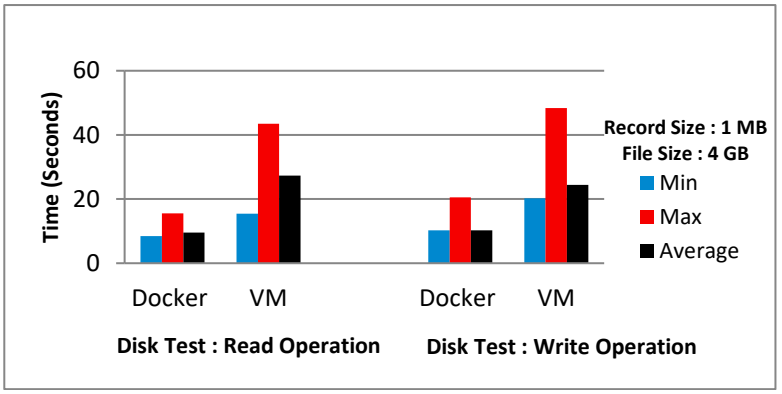

Fig. 7. IOzone Benchmark Disk Performance

4.4 Load Testing

For the load testing performance comparison Apache Benchmark tool is used, in which it measures the number of requests per second a given system can tolerate. A python program is executed to test the load using the Apache Benchmarking tool. Fig. 8. shows that the throughput analysis for VM is much lesser compared to that of Docker. This is because of higher network latency in Virtual machine than in Docker. The analysis shows that Docker container is better than the virtual machine in handling the number of requests per second.

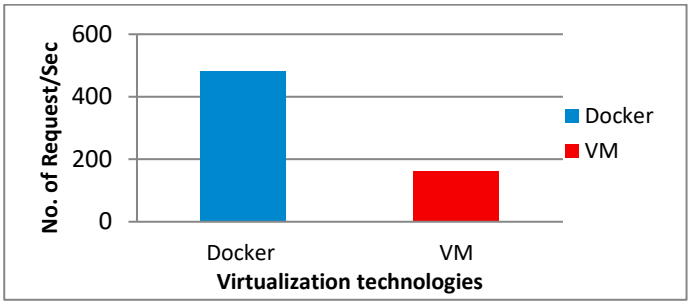

Fig. 8. Load test comparison between Docker and Virtual Machine

4.5 Operation speed measurement

The eight queen's problem places eight queens on an 8×8 chessboard such that none of them attack one another. The test measures how long it takes to solve the problem. The Eight queen program is written in python and determines the system's computational performance. Fig. 9. shows the computational performance of both docker and virtual machines. Based on the execution time, the docker container takes less to solve the problem, whereas the virtual machine takes a much longer time.

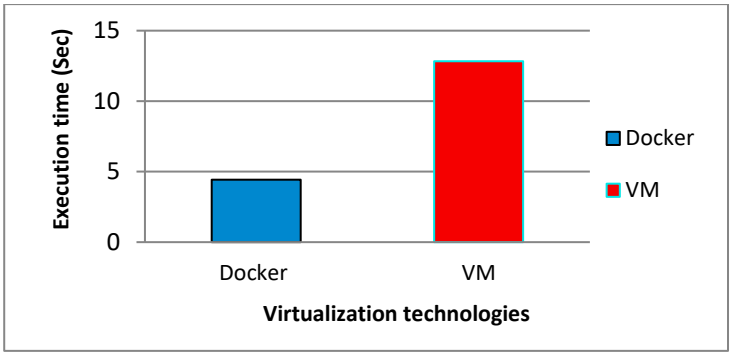

Fig. 9. Eight Queen Program Performance Comparison

Eight Puzzle test: Taking 4×4 board with 8 tiles and one empty space. Using the empty space, the arrangement of the number of tiles to match the final configuration is the main objective. It can slide four adjacent operations (right, left, below, and above) tiles into the empty space — *the* test measures how long it takes to solve the problem. The Eight puzzle program is written in python and determines the system's computational performance. Fig. 10. shows the computational performance of both docker and virtual machines. Based on the execution time, docker container takes less to solve the problem, whereas the virtual machine takes a much longer time.

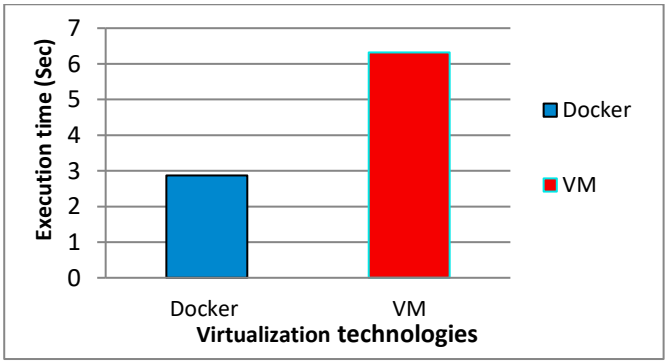

Fig. 10. Eight puzzle test performance comparison

4.6 t-tests Analysis

A t-test statistical measurement is used to decide whether there is a critical distinction between the methods for two groups, which might be connected in the related feature. In the following presentation of the results, statistical inference technique t-test has been used to prove that docker container significantly outperforms virtual machine. Moreover, the confidence level for the threshold values is taken as α of 0.05. The probability of two sets of data can be determined by taking t-statistic, t-distribution values, and the degree of freedom.

A null hypothesis (H0) and the alternative hypothesis (H1) conventions are used in the analysis part. Test analysis is carried out with consideration of the null hypothesis as true, an assumption for further statistical analysis. The resultant of the test may then prove that the assumption is probably wrong if H0 is true.

Table 2. Statistical Analysis

T-test Requirements	**Results**
H_0(Null Hypothesis)	$Docker_{time}$ = Virtual $Machine_{time}$
H_1(Alternative Hypothesis)	$Docker_{time}$ ≤Virtual $Machine_{time}$
Alpha(ά)	0.05
No of samples(N)	10
∑x	879
$\sum d^2$	7275.4
$\bar{x}$	87.9
Standard Deviation(s_x)	85.29
Test statistic(t)	-2.43
μ_L	-2.26

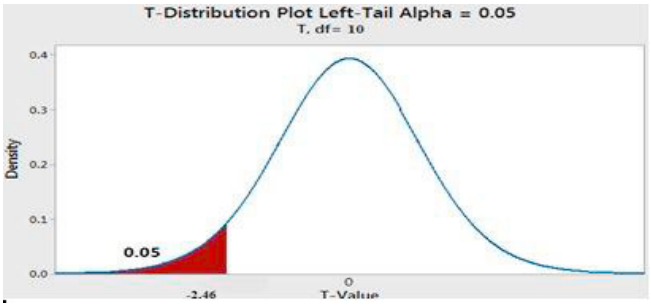

Fig. 11. left tailed t-test curve

It was required to test whether the docker container takes less time to execute the operations than a virtual machine. The sample test consists of 10 experiments on the operation of finding the prime numbers. Table 3 shows the 10 samples taken for the analysis part. The mean amount of time taken for performing the operation of the virtual machine was found to be 153.5 seconds. Now it is required to check whether the Mean amount of time taken with docker container is less than a virtual machine. The equation for t statistics (t) is

$$t = \frac{x-\mu}{S/\sqrt{n}}$$

Table 2. describes the statistical test summary as we can clearly see that the lower confidence limit(μL) is less than test statistic(t), Since -2.89 < -2.26 So, therefore, H0 is rejected H1 is accepted. From Fig 11, we can see that test statistic (t) lies in the rejection region. Hence it could be said that the amount of time taken by the virtual machine is more than the docker container to perform the operation of finding the prime numbers; it means H0 is rejected.

Table 3. describes the execution time of the virtual machine and docker with respect to the maximum prime number calculation. To perform this operation, we have used different test values. These 10 samples are considered as

execution time at different input values, where N represents the number of values given for the calculations. The program for finding the prime number at a given set of values are executed on a virtual machine and docker installed machine.

Table 3. Experimental Results

Test Input N	Virtual Machines Prime number program (execution time in seconds)	Docker Containers Prime number program (execution time in seconds)
10	13	8
20	47	20
30	59	35
40	88	53
50	120	72
60	144	92
70	188	115
80	225	140
90	265	164
100	386	180

5. Conclusion

Docker Container is an emerging lightweight virtualization technology. This work evaluates two virtualization technologies namely docker containers and virtual machines. Performance evaluation is carried out on virtual machine and Docker container-based hosts in terms of CPU Performance, Memory throughput, Disk I/O, Load test, and operation speed measurement. It is observed that Docker containers perform better over VM in every test, as the presence of QEMU layer in the virtual machine makes it less efficient than Docker containers. The performance evaluation of both containers and the virtual machine is performed with the usage of benchmarking tools such as Sysbench, Phoronix, and apache benchmarking.

As future work, we plan to work on the scheduling of containers in docker also to work on a more secure variant of containers which will reduce security constraints.

References

[1] Docker. https://docs.docker.com/ 2019 [Online; Accessed 24-03-2019]
[2] Shivaraj Kengond, DG Narayan and Mohammed Moin Mulla (2018) *"*Hadoop as a Service in OpenStack" *in Emerging Research in Electronics, Computer Science and Technology* , pp 223-233.
[3] C. G. Kominos, N. Seyvet and K. Vandikas, (2017) "Bare-metal, virtual machines and containers in OpenStack" *20th Conference on Innovations in Clouds, Internet and Networks (ICIN), Paris,* pp. 36-43.
[4] Higgins J., Holmes V and Venters C. (2015) "Orchestrating Docker Containers in the HPC Environment*". In: Kunkel J., Ludwig T. (eds) High Performance Computing. Lecture Notes in Computer Science,* vol 9137.pp 506-513
[5] Max Plauth, Lena Feinbube and Andreas Polze, (*2017)* "A Performance Evaluation of Lightweight Approaches to Virtualization", *CLOUD COMPUTING: The Eighth International Conference on Cloud Computing, GRIDs, and Virtualization.*
[6] Kyoung-Taek Seo, Hyun-Seo Hwang, Il-Young Moon, Oh-Young Kwon and Byeong-Jun Kim "Performance Comparison Analysis of Linux Container and Virtual Machine for Building Cloud," *Advanced Science and Technology Letters* Vol.66, pp.105-111.
[7] R. Morabito, J. Kjällman and M. Komu, (2015) "Hypervisors vs. Lightweight Virtualization: A Performance Comparison", *IEEE International Conference on Cloud Engineering, Tempe, AZ*, pp. 386-393.
[8] Babak Bashari Rad, Harrison John Bhatti and Mohammad Ahmadi (*2017)* "*An Introduction to Docker and Analysis of its Performance*" IJCSNS International Journal of Computer Science and Network Security, VOL.17 No.3, pp 228-229.
[9] Kozhirbayev, Zhanibek, and Richard O. Sinnott. (2017) "A performance comparison of container-based technologies for the cloud", *Future Generation Computer Systems,* pp: 175-182.
[10] Felter, Wes, et al. (*2015)* "An updated performance comparison of virtual machines and linux containers", *IEEE international symposium on performance analysis of systems and software (ISPASS)*. pp:171-172.
[11] Sysbench. https://wiki.gentoo.org/wiki/Sysbench 2019 [Online; Accessed 24-03-2019]
[12] Phoronix Benchmark tool. https://www.phoronix-test-suite.com/ 2019 [Online; Accessed 24-03-2019]
[13] Apache Benchmark tool. https://www.tutorialspoint.com/apache_bench/ 2019 [Online; Accessed 24-03-2019]