Article **Journal of Scientific and Technical Applications**
December 2021, Vol.7 No.20 1-9

Containers vs. virtual machines: performance comparison

Contenedores frente a máquinas virtuales: comparación de rendimiento

RAMIREZ-PERALTA, David† & ALCUDIA-FUENTES, Ever

TecNM campus Comalcalco, Mexico.

ID 1er Autor: *David, Ramírez-Peralta* / **ORC ID:** 0000-0003-3181-1351

ID 1st Co-author: *Ever, Alcudia-Fuentes* / **ORC ID:** 0000-0001-6432-7967

DOI: 10.35429/JSTA.2021.20.7.1.9 Received July 10, 2021; Accepted December 30, 2021

Abstract

Server virtualization is a technological innovation widely used in information technology (IT) companies. Virtualization provides a platform to run different operating system services in the cloud. It makes it easy to build multiple virtual machines on a single basic physical machine, either in the form of hypervisors or containers. To host many microservices applications, emerging technology has introduced a model consisting of different operations performed by smaller individual deployed services. Therefore, the demand for low-cost virtualization techniques is developing rapidly. There are many lightweight virtualization technologies; Docker is one of them, which is an open-source platform. This technology enables developers and system administrators to build, create, and run applications using the Docker engine. This document provides performance evaluation of Docker containers and virtual machines using standard benchmark tools such as Sysbench, Phoronix, and Apache, including CPU performance, memory performance, storage read/write performance, load test and measurement of operating speed.

Virtualization, Docker containers, Virtual Machines, Benchmark tools

Resumen

La virtualización de servidores es una innovación tecnológica ampliamente utilizada en empresas de tecnologías de la información (TI). La virtualización proporciona una plataforma para ejecutar diferentes servicios de sistemas operativos en la nube. Facilita la construcción de múltiples máquinas virtuales en una sola máquina física básica, ya sea en forma de hipervisores o contenedores. Para alojar muchas aplicaciones de microservicios, la tecnología emergente ha introducido un modelo que consta de diferentes operaciones realizadas por servicios implementados individuales más pequeños. Por lo tanto, la demanda de técnicas de virtualización de bajo costo se está desarrollando rápidamente. Existen muchas tecnologías de virtualización ligeras; Docker es uno de ellos, que es una plataforma de código abierto. Esta tecnología permite a los desarrolladores y administradores de sistemas construir, crear y ejecutar aplicaciones usando el motor de Docker. Este documento proporciona la evaluación del rendimiento de los contenedores Docker y las máquinas virtuales utilizando herramientas de referencia estándar como Sysbench, Phoronix y Apache, que incluyen el rendimiento de la CPU, el rendimiento de la memoria, el rendimiento de lectura/escritura del almacenamiento, la prueba de carga y la medición de la velocidad de operación.

Virtualización, Contenedores Docker, Máquinas virtuales, Herramientas Benchmark

Citation: RAMIREZ-PERALTA, David & ALCUDIA-FUENTES, Ever. Containers vs. virtual machines: performance comparison. Journal of Scientific and Technical Applications. 2021. 7-20:1-9.

† Researcher contributing as first author.

Introduction

In recent years, the trend to cloud computing is increasing. Numerous technologies are developed by the IT industry 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, the applications and dependencies must be organized and isolated. Due to virtualization, multiple applications can run on the same physical hardware.

The drawbacks of virtualization techniques are: virtual machines are generally large, with unstable performance due to running multiple virtual machines, the boot process takes a long time to run, and virtual machines cannot resolve difficulties such as manageability, software updates, and continuous integration / delivery. These problems led to the emergence of a new process called containerization that further led to virtualization at the operating system level, while virtualization brings abstraction at the hardware level. Containerization uses the operating system of hosts that share relevant libraries and resources. It is more efficient because there is no guest operating system. In the host kernel, application-specific binaries and libraries can be processed, making execution very fast. Containers are created with the help of the Docker platform, which combines applications and their dependencies. These containers always run on top of the operating system kernel in sandbox. This Docker containerization feature ensures that the environment supports any related applications (Docker, 2021).

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

Background

Docker

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

Docker Engine

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

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

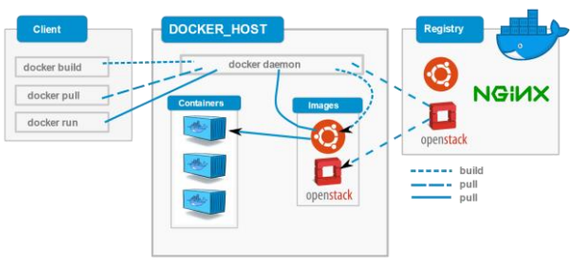

Figure 1 Docker Engine Architecture
Source: https://docs.docker.com/get-started/overview/

Docker Client-Server

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

Docker Image

Docker images can be created using two methods. The main method is to create an image with the help of a model.

The template consists of base images, be it an OS like Centos, Debian, Ubuntu 20.04, Fedora, or any other lightweight base OS image. Generally, the base images are the base of each image.

A new image needs to be created each time base images are created from scratch. This type of creating a new image is called "making a change".

The next method is to create a docker file that has all the instructions for creating a docker image. When the docker build command is run from the terminal, the image will be created with all the dependencies mentioned in the docker. This process is known as an automated method of creating an image.

Docker containers

Docker containers are created by the Docker image. To run the application in a confined manner, all the kits required for the application must be in the container. Container images can be created based on the service requirements for the application or software. Suppose that an application that includes the Ubuntu operating system and the Nginx server needs to be added to the Docker archive. Using the "docker run" command, a container is created with the Ubuntu operating system image consisting of the Nginx server and starts running.

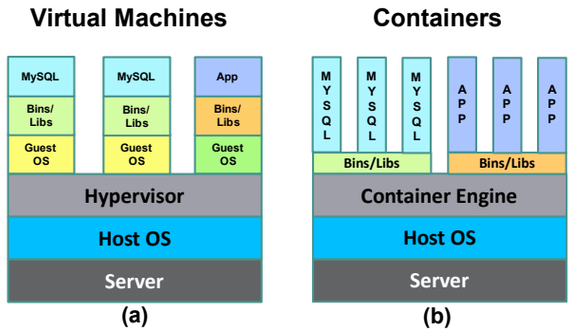

Figure 2 (a) Hypervisor-based architecture (b) Container-based architecture
Source:https://www.freecodecamp.org/news/comprehensive-introductory-guide-to-docker-vms-and-containers-4e42a13ee103/

Comparison between virtual machine and Docker container

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

	Docker Container	Virtual Machine
Isolation process level	Operating system	Hardware
Operating System	Shared	Separated
Boot time	Short	Long
Use of resources	Less	Higher
Pre-build images	Available for home server	Difficult to find and manage
Custom preset images	Easy to build	Difficult to build
Size	Smaller only with the Docker Engine on top of the host operating system	Larger because it contains the entire underlying operating system
Mobility	Destroyed and recreated instead of moving	Easy to switch to a new host operating system
Creation time	In seconds	Long

Table 1 Docker containers vs virtual machine.
Source: Self-made

ISSN 2444-4928

Related work

The use of virtual machines in development (VM) is common in organizations. Virtual machines often run complex tasks like Hadoop (Kengon *et al*, 2019). However, users also use virtual machines even to start a small application, which makes the system inefficient. It is necessary to launch a lightweight application, which is faster and makes the system efficient. The Docker container is one of the technologies that offers lightweight virtualization, and this motivates us to do the work in the background. Figure 1 shows the architecture of the Docker Container.

In (Kominos *et al*. 2017), the authors present an overview of Docker container and virtual machine performance evaluation in terms of CPU performance, memory performance, disk I/O, and operational speed measurement. The authors in (Higgins *et al*, 2015) focused on implementing Docker containers on HPC Cluster. In the last part of the document, the authors explain the different implementation approaches to choosing the container model and the use of LNPACK and BLAS. In (Plauth *et al*,2017), the authors discuss lightweight virtualization approaches, which address container and unikernel issues. In addition, the paper also discusses the statistical evaluation of the ANOVA test and a post-hoc comparison using Tukey's method of the collected data. The author also discusses the different benchmarking tools used for unikernel and container comparison. The static HTTP server and key value store parameters are used for experimental analysis of application performance, which is deployed in the cloud. The Nginx server is used for HTTP performance, and to measure get and configure operations, the Redis benchmark is used.

The authors of (Chae *et al*, 2019) discuss evaluation with benchmarking applications using KVM, Docker, and OSv. In (Morabito *et al*, 2015), the authors discuss a short survey on virtual machines and containerized technologies. It also looks at docker and docker performance with various parameters CPU, memory performance, disk I / O. In (Babak *et al*, 2017), the author discusses the fundamental concepts of docker architecture, docker components, docker images, docker logs, docker client, and docker architecture. server.

The difference between virtual machines and docker container is discussed. In (Kozhirbayev and Sinnott, 2017), the authors explain the performance comparison of container-based technologies for the cloud with a different set of parameters. The document provides information on the use of OpenStack-based cloud implementations, which are considered for comparison. The platforms used for performance comparison are docker, LXC, and flockport. The authors in (Pérez, 2021) discuss the virtual machine and docker performance comparison against various parameters such as CPU, network, disk, and two real server applications, which are Redis and MySql.

Methodology

In this section, KVM and Docker evaluation is done using benchmarking tools. The next benchmarking tools used for performance evaluation are Sysbench (Kopytov, 2021), Phoronix (Larabel and Tippett, 2021) and Apache benchmark (The Apache Software Foundation, 2021).

These benchmark tools measure CPU performance, memory performance, storage reads and writes, load testing, and operating speed measurement. The two HP servers are used for performance evaluation in relation to various parameters, for this server it is used as a virtual machine that is installed on a host or host operating system (Windows 10) and while the guest operating system (Ubuntu 20.04) with the addition of this Docker engine it is installed on top of the virtual machine and another server like bare metal with the Ubuntu 20.04 host operating system and the Docker engine installed on it. All tests were performed on an HP server with two 2.40 GHz Intel Xenon E5-2620 v3 processors with a total of 12 cores and 64 GB of RAM. 64-bit Ubuntu 20.04 with Linux kernel 3.10.0 was used for all testing. For consistency and consistency, the same operating system, Ubuntu 20.04, was used as the base image for all of the Docker containers. The 12vCPUs and sufficient RAM are configured for VMs. Figure 3. presents the evaluation methodology of different virtualization technologies with various benchmarking tools.

ISSN 2444-4928

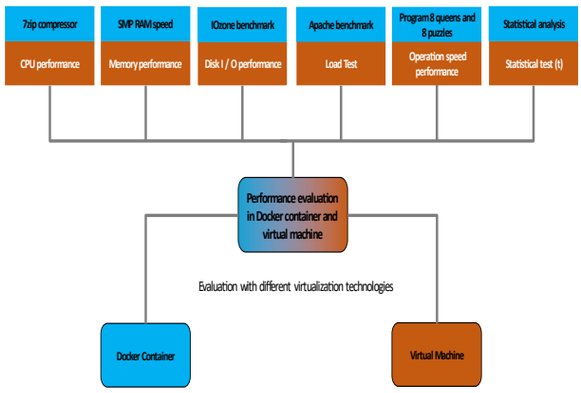

Figure 3 Performance evaluation with various benchmarking tools
Source: Self-made

Results and discussions

This section discusses the performance analysis of virtualization technologies. The results are classified into four subsections. Section 4.1 describes all measures of CPU, memory performance; The storage, reading and writing measurements are shown in sections 4.2 and 4.3. Section 4.4 presents a load test analysis. Section 4.5 describes the measurement of operating speed, which consists of two tests: eight queen problem and eight puzzle problems. Finally, in section 4.6, a statistical analysis t-test is performed.

CPU performance

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

Maximum prime number operation

In the test of the Sysbench tool performed to find out the time taken to realize the maximum prime number. The maximum prime number value for the operation is taken as 50,000 with a time of 60 seconds and 4-wire operations. In Fig. 4. you can see that the docker container takes much less time to execute the operation compared to the VM. This is due to the presence of the hypervisor in the virtual machine. Therefore, the execution takes more time.

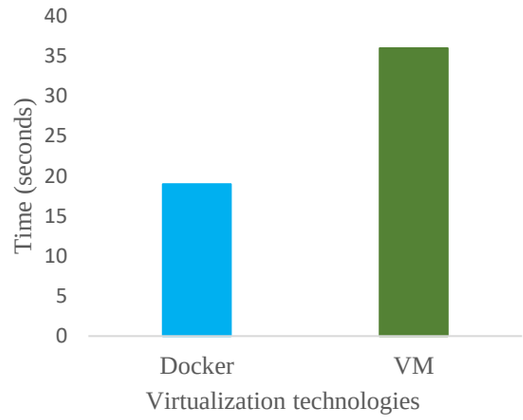

Figure 4 CPU comparison. Docker vs virtual machine
Source: Self-made

7-zip compression test

7-Zip is an open-source file compression software, which is a service for compressing a group of files in containers known as "archives". The two tests for the LZMA benchmark are the LZMA compression and decompression methods. This test measures the time it takes to compress a file using the 7 Zip file compressor. The file size used for the compression test is 10 GB. Figure 5 shows the CPU performance comparison with the 7 Zip compression test. Therefore, according to the results obtained, the performance of the Docker container is much better than that of the VM when it is done with a large amount of file compression.

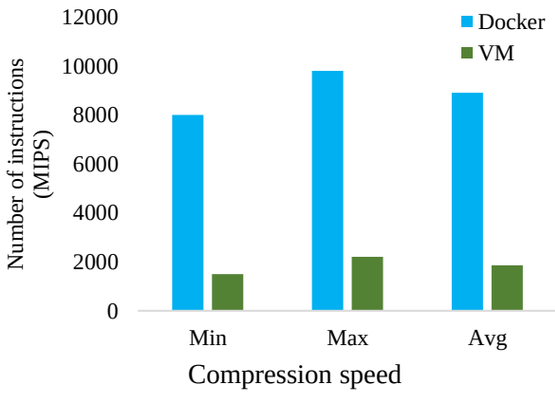

Figure 5 Compression test for CPU performance
Source: Self-made

Memory performance

The SMP RAM Speed Test (Symmetric Multiprocessing) is a memory and cache benchmarking tool used to measure RAM speed for virtualization technologies, i.e. Docker and Virtual Machine. Fig. 6. depicts RAM speed comparisons between virtualization technologies. The next two main parameters are considered when testing RAM speed. The INTmark and FLOATmark components are used in the RAM Speed SMP benchmarking tool, which measures the maximum possible memory and cache performance when reading and writing individual blocks of data.

INTmem and FLOATmem are synthetic simulations but closely balanced with the real world of computing. Each consists of four subtests (Copy, Scale, Add, Triad) to measure different aspects of memory performance. Data transfer from one memory location to another is done by the copy command, that is (X = Y). The modification of the data before writing is multiplied by a certain constant value that is carried out by the scale command, that is (X = n * Y). Data is read from the first memory location and then from the second when ADD commands are called. Then the resulting data is placed third (X = Y + Z). Triad is a combination of Add and Scale. The data is read from the first memory location to scale and then added from the second place to write it to the third place (X = n * Y + Z). Figure 6. shows memory performance relative to the SMP RAM speed test.

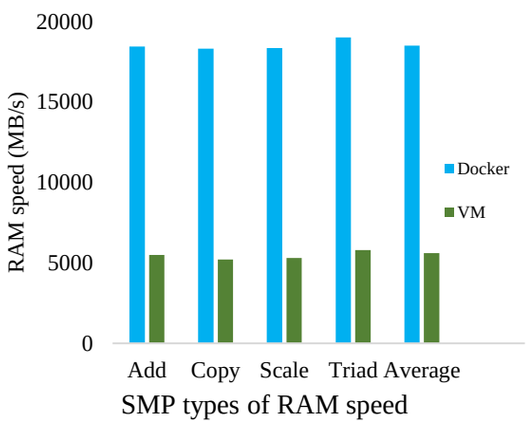

Figure 6 RAM speed comparisons between virtualization technologies
Source: Self-made

Disk read/write performance

To test the performance of the hard drive, the IOzone benchmark tool is used for performance analysis. To test operations such as system read and write, a log size of 1 MB and a file size of 4 GB were used. From figure 7 it can be inferred that the performance of Docker is much better compared to the virtual machine. The disk read and write operations of a virtual machine are reduced by more than half that of the Docker container (approximately 54%). Fig. 7. shows the disk performance of both virtual machines and Docker.

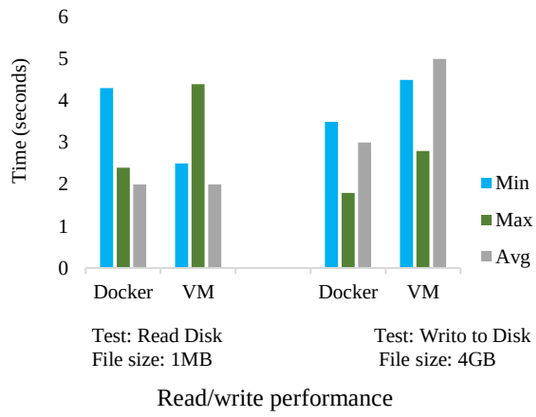

Figure 7 Disk performance with IOzone Benchmark.
Source: Self-made

Load Test

The Apache Benchmark tool is used to compare the performance of the load tests, in which it measures the number of requests per second that a given system can tolerate. A Python program is run to test the load using the Apache Benchmarking tool. Figure 8. shows that VM performance analysis is much lower compared to Docker container.

This is due to higher network latency on the virtual machine than on Docker. The analysis shows that the Docker container is better than the virtual machine in handling the number of requests per second.

ISSN 2444-4928

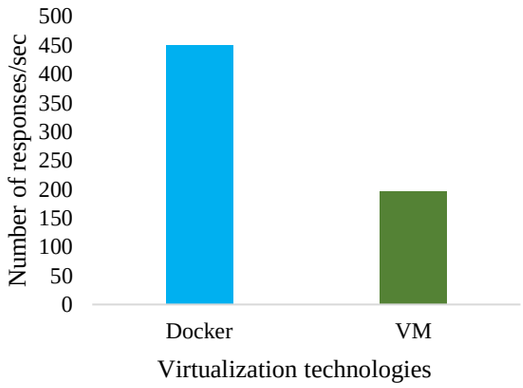

Figure 8 Load test comparison between Docker and virtual machine
Source: Self-made

Measuring operating speed

The eight queens problem places eight queens on an 8 × 8 chessboard so that none attack each other. The test measures how long it takes to solve the problem. The 8Reinas program is written in Python and determines the computational performance of the system. Fig. 9. shows the computational performance of both the docker and virtual machines. Depending on the runtime, the docker container takes less time to resolve the problem, while the virtual machine takes much longer.

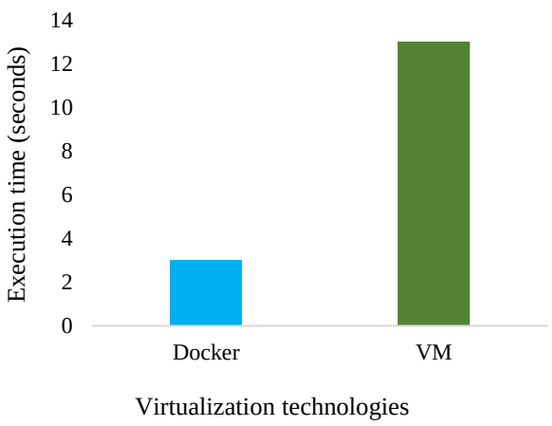

Figure 9 8 queens program performance comparison
Source: Self-made

Test of eight puzzles: Taking a 4 × 4 board with 8 tiles and an empty space. Using the empty space, arranging the number of tiles to match the final configuration is the main goal. You can slide four adjacent feature tiles (right, left, down, and up) in the empty space; the test measures how long it takes to solve the problem.

The 8breaker program is written in Python and determines the computational performance of the system. Figure 10. shows the computational performance of both the docker and the virtual machines. Depending on the runtime, the docker container takes less time to resolve the problem, while the virtual machine takes much longer.

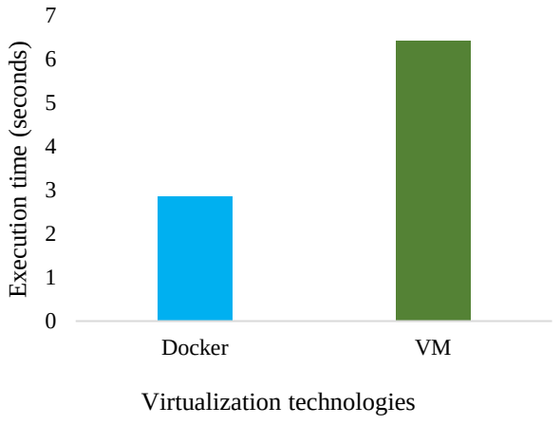

Figure 10 Comparison of program 8 puzzle performance.
Source: Self-made

t-test analysis

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

In the analysis part, the null hypothesis (H_0) and alternative hypothesis (H_1) conventions are used. The test analysis is carried out considering the null hypothesis as true, an assumption for further statistical analysis. The test result can prove that the assumption is probably wrong if H_0 is true.

ISSN 2444-4928

T-test requirements	Results
H_0	$Docker_{time}$=Virtual $Machine_{time}$
H_1	$Docker_{time}\leq$Virtual $Machine_{time}$
Alfa(α)	0.05
Number of examples (N)	10
∑x	879
$\sum d^2$	7275.4
x	87.9
Standart deviation	85.29
Statistical test (t)	-2.43
µL	-2.26

Table 2 Statistical analysis
Source: Self-made

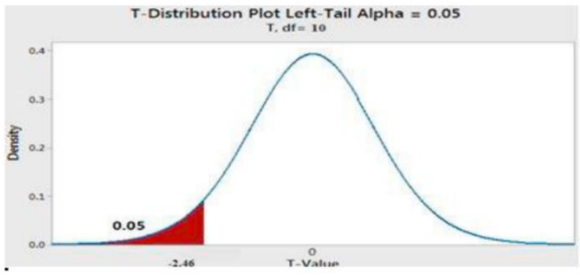

Figure 11 Left-tailed t-test curve
Source: Self-made

It was required to test if the docker container takes less time to execute 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 average amount of time required to perform the virtual machine operation was found to be 153.5 seconds. Now you need to check if the average amount of time spent with the docker container is less than a virtual machine. The equation for statistical t (t) is:

$$t=x-\mu$$

$$S/\sqrt{n}$$

Table 2 describes the statistical summary of the test, since we can clearly see that the lower confidence limit (µL) is lower than the test statistic (t). Since -2.89 <-2.26 So, therefore, H_0 is rejected H_1 is accepted. In Fig. 11, we can see that the test statistic (t) is in the rejection region. Hence, it can be said that the amount of time it takes the virtual machine is greater than the docker container to perform the operation of finding the prime numbers; means that H_0 is rejected.

Table 3 describes the runtime of the virtual machine and docker with respect to calculating the maximum prime number. To perform this operation, we have used different test values. These 10 samples are considered as running time at different input values, where N represents the number of values given for the calculations. The program to find the prime number in a given set of values runs in a virtual machine and in a docker container.

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

Table 3 Experimental results
Source: Self-made

Conclusions

Docker Container is an emerging lightweight virtualization technology. This paper evaluates two virtualization technologies, namely, Docker containers and virtual machines. Performance evaluation is carried out on virtual machines and hosts based on Docker containers in terms of CPU performance, memory performance, disk I / O, load testing, and operational speed measurement. It is observed that Docker containers perform better on VM in each test, since the presence of the QEMU layer in the virtual machine makes it less efficient than Docker containers. Performance evaluation of both the containers and the virtual machine is done using benchmarking tools such as Sysbench, Phoronix, and apache benchmarking. As future work, we plan to work on container scheduling in Docker as well to work on a more secure variant of containers that will reduce security restrictions.

ISSN 2444-4928

References

Babak Bashari, R., Harrison, J. B. and Mohammad A. (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.

Chae, M., Lee, H. and Lee, K. (2019). A performance comparison of linux containers and virtual machines using Docker and KVM. Cluster Comput 22, 1765–1775. https://doi.org/10.1007/s10586-017-1511-2.

Docker. [Online] https://docs.docker.com/get-started/overview/ Accessed: 22-03-2021

Evens Cubillos, E. J. (2021). Saldo y disponibilidad en la nube para navegación 2.0 (Doctoral dissertation, Universidad Andrés Bello).

Higgins J., Holmes V. and Venters C. (2015) Orchestrating Docker Containers in the HPC Environment. In: Kunkel J., Ludwig T. (eds) High Performance Computing. ISC High Performance 2015. Lecture Notes in Computer Science, vol 9137. Springer, Cham. https://doi.org/10.1007/978-3-319-20119-1_36

Kengond S., Narayan D.G., Mulla M.M. (2019) Hadoop as a Service in OpenStack. In: Sridhar V., Padma M., Rao K. (eds) Emerging Research in Electronics, Computer Science and Technology. Lecture Notes in Electrical Engineering, vol 545. Springer, Singapore. https://doi.org/10.1007/978-981-13-5802-9_21.

Kominos, C. G., Seyvet, N. and Vandikas, K. (2017). Bare-metal, virtual machines and containers in OpenStack. 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), Paris, pp. 36-43. https://doi.org/10.1109/ICIN.2017.7899247.

Kopytov, A. (2020). Sysbench (Version 1.0.20). [Software de computador]. https://github.com/akopytov/sysbench/

Kozhirbayev, Z. and Sinnott, R.O. (2017). A performance comparison of container-based technologies for the cloud, Future Generation Computer Systems, pp: 175-182. https://doi.org/10.1016/j.future.2016.08.025

Morabito, R., Kjällman, J. and Komu, M. (2015). Hypervisors vs. Lightweight Virtualization: A Performance Comparison, 2015 IEEE International Conference on Cloud Engineering, pp. 386-393, doi: 10.1109/IC2E.2015.74.

Pérez Znakar, O. P. (2021). Computación de Altas Prestaciones en entornos contenerizados.
Larabel, M. and Tippett, M. (2021). Phoronix Test Suite (Version 10.2) [Software de computador] Phoronix. https://www.phoronix-test-suite.com/

Plauth, M., Feinbube, L., and Polze, A. (2017). A Performance Evaluation of Lightweight Approaches to Virtualization. CLOUD COMPUTING: The Eighth International Conference on Cloud Computing, GRIDs, and Virtualization.

Pons, V., Morinelli, M., & Barcia, E. (2021). Integración de una Service Mesh a una plataforma de integración basada en microservicios.

Rodriguez Dominguez, J. (2021). Serverless Computing, Funciones como Servicio (FaaS) para el soporte de cargas computacionales en la nube.

Santa Rendón, D. (2021). Aplicación de una arquitectura basada en Service Mesh para una plataforma cognitiva utilizando Kubernetes e Istio.

The Apache Software Foundation. (2021). Apache HTTP Server benchmarking tool (Versión 2.4) [Software de computador]. https://httpd.apache.org/docs/2.4/programs/ab.html.

ISSN 2444-4928