A performance evaluation between Docker container and Virtual Machines in cloud computing architectures

Luis Herrera-Izquierdo, Marc Grob

Escuela de Sistemas y Computación, Pontificia Universidad Católica del Ecuador Sede Esmeraldas, Ecuador.

Autor para correspondencia: luis.herrera@pucese.edu.ec

Fecha de recepción: 17 de mayo 2017 - Fecha de aceptación: 7 de agosto 2017

ABSTRACT

Reliability, portability, scalability and availability of applications are essential features of cloud computing in the software architecture of enterprises, that usually makes use of virtual machines (VM's). The hardware resources of cloud computing are always limited, for this reason it is important that the available resources are adequately allocated to obtain the best possible performance. The container technology is an alternative to VM as it allows to virtualize operating systems, package applications along with the required dependencies and deploy them as an instance of the operating system, permit applications to run independently, and consume only the necessary resources. This article, related to the area of parallel and distributed computing, presents a performance evaluation analyzing several aspects between VM's and Docker containers, based on different benchmark tools. The test configuration is based on the principles of high performance computing (HPC), adapting the same setup used to measure the performance of the processing of big amount of data or information, demanding all out of the available hardware resources. The results of the tests reveal that Docker containers perform better compared to VM's based on VirtualBox.

Keywords: Container, cloud computing, HPC, virtual machine, performance testing.

RESUMEN

La confiabilidad, portabilidad, escalabilidad y disponibilidad de las aplicaciones, son características esenciales que permiten implementar *computación en la nube*, usualmente basada en máquinas virtuales (VM's), en una arquitectura de software empresarial. Los recursos de hardware de la *computación en la nube*, son siempre limitados. Por esto, y para obtener el mejor rendimiento posible, es importante que los recursos disponibles sean consumidos adecuadamente. La tecnología de contenedores representa una alternativa frente a las VM's, debido a que nos permite virtualizar sistemas operativos, que empaquetan aplicaciones junto con las dependencias necesarias y las despliega en una instancia del sistema operativo. Esto permite que las aplicaciones se ejecuten de manera independiente y consuman solamente los recursos necesarios. En este artículo, relacionado con el área de computación paralela y distribuida, se presenta una evaluación del rendimiento computacional de dos tendencias de virtualización: VM's y *contenedores Docker*, mediante el uso de varias herramientas tipo *benchmark*. Las pruebas se realizaron acorde a los principios de computación de alto rendimiento (HPC): similar a una evaluación de rendimiento para grandes cantidades de datos y procesamiento de información, esto es, implicando una alta demanda de los recursos de hardware disponibles. Los resultados de las pruebas demuestran que los *contenedores Docker* tienen un mejor rendimiento frente a las VM's basadas en VirtualBox.

Palabras clave: Contenedor, computación en la nube, HPC, máquina virtual, pruebas de rendimiento.

1. INTRODUCTION

Software development using virtual machines (VM's) allows to isolate dependencies of software libraries and limit the resources needed for a system to work correctly. Most importantly it has simplified the portability and scalability (Xavier, Ferreto & Jersak, 2016). VM's are operated by a hypervisor on which an operating system is installed (Kleyman, 2012). Using multiple controllers and sharing hardware resources with the host computer enables to run various VM's on one Server.

VM's are widely used in Infrastructure as a Service (IaaS) architectures for the facilities that they offer to administer resources. VM's are playing a vital role in cloud computing (Peter Mell, 2011). Large computer companies like Google, Microsoft or Amazon have built different types of services in the cloud offering customizable VM's. These types of services offer adjustable hardware components, which are configured depending on the requirements (Barik, Lenka, Rao & Ghos, 2016).

Container technology is an alternative to VM's; it improves the use of hardware recourses, and allows to share the consumption of CPU, RAM, and data transfer via the network. The same hardware resources can be used by various software applications. This opens the possibility to extend de Infrastructure as a Service (IaaS) concept by introducing Containers as a Service (CaaS). The container technology is rapidly growing because of its characteristics that allow to run even more software applications on the same hardware than using VM.

Docker (Docker, 2017) is an open source project that provides a high-level API to create and provide images of containers. The main difference between Docker containers and VM's is that several containers can use or access the same resources, because they use the same host kernel unlike the hypervisor of a VM that uses different instances of a kernel for each one (Jacobsen & Canon, 2015). Docker is running on a host machine sharing the kernel functionality, which can deploy the necessary resources and then isolate the containers with its own layer of protection. The container approach allows more efficient use of available resources versus VM (Preeth, Mulerickal, Paul & Sastri, 2015). This research measures and compares the performance of VM's and containers using a mix of benchmark tools for performance testing based on the investigations of Chung, Quang-Hung, Nguyen & Thoai (2016) and Morabito (2016).

The tools used to test the performance are all open source and commonly used to measure the performance of HPC architectures, as well as in benchmark tests, among other, supercomputing clusters.

This research took place in an educational environment and is structured as follows: in Section 2 the tools used for the development of research are described, an analysis of the obtained results is presented in Section 3, and the most relevant conclusions are summarized in Section 4.

2. METHOD

Docker is an alternative in the field of cloud computing. As shown in Table 1 it can handle the concept of services but has more advantages (Preeth *et al.*, 2015).

Table 1. VM vs Docker.

Virtual Machine	Docker
Need of an operating system for each service	Shares system components form the host; does not need an operating system for each service
Slow startup and shutdown	Fast startup and shutdown
Share files with the Host via the Hypervisor. Direct access is not possible.	Share files with the host using Secure Copy (SCP)
Data transfer network over Hypervisor form bottlenecks	Creates a network bridge for each service which allows to use all the bandwidth.
Consumes all RAM assigned to the VM	Consumes only RAM needed from the service

One of the advantages of Docker is that a service can be started and turned off faster than when using a VM (Spoiala, Calinciuc & Turcu, 2016). This is possible because the Docker engine is linked to the components of the host computer while the VM's depend on a hypervisor (Fig. 1). The latter needs several drivers to provide functionality to VM's with an impact on performance.

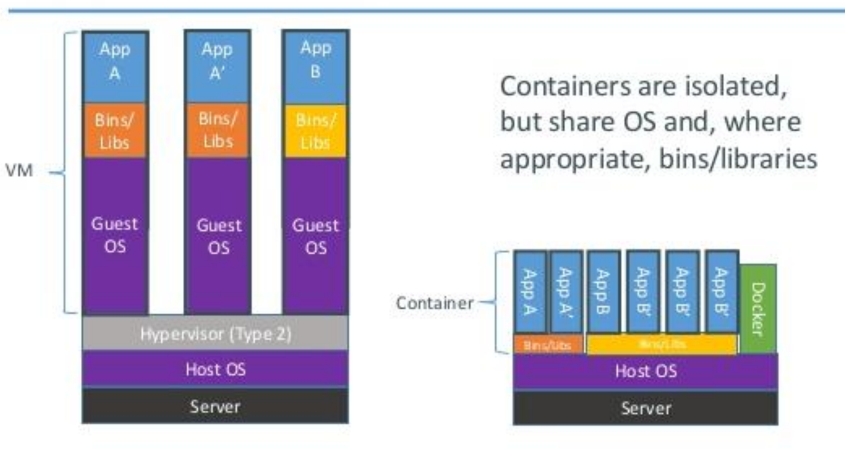

Figure 1. Virtual machine vs Docker (Docker, 2017).

Virtualbox was chosen as VM engine, mainly because it is licensed with GPL, is freely available and for its graphical user interface that simplifies the handling. As base operating system for both, VirtualBox and Docker, a 64-bit version of Debian 8 was used. The tests were performed on the 64-bit version of VirtualBox 5.1.6, without installing the extension pack. The Docker engine 1.12 was used for the container environment.

Several different software libraries are available to measure the performance of the container and the VM. In this study, the High-Performance Computing Linpack Benchmark (HPL) (Petitet, Whaley, Dongarra & Cleary, 2016) was used to measure the CPU performance. HPL is a software library tool that measures the floating-point computing performance applying linear arithmetic equations. The efficiency of RAM was measured with sysbench 0.4.12 (Pipes, 2010); iperf (Parziale, Louie, Marins, Santos & Venkatesan, 2012) was used to measure the network bandwidth; and with Bonnie++ (Coker, s.f.), a benchmark tool for filesystems, the read and write performance was evaluated.

The tests were executed on a HP Proliant server with a single Intel Xeon E5-2407 @ 2.20 GHz processor with 16 GB RAM, 1 TB Hard Disk and a Gigabit LAN (10/100/1000) network card. As host operating system, the 64-bit version of Debian 8 was used, for both: the virtualization and the container environment.

2.1. HPL configuration

To maximize the use of resources, given their availability, HPL requires a specific configuration (Sindi, 2009). The size of the problem N, the block size NB, the number of processors P, and the number of cores available Q, are the variables required to configure. P is 1, because we only dispose of one CPU with 4 cores, which is the value for Q. N can be calculated with the following equation (1):

$$N = \sqrt{\left(\frac{TMG * 1024 * 1024 * 1024 * NN}{8}\right)} * PM \tag{1}$$

$$N = \sqrt{\left(\frac{12 * 1024 * 1024 * 1024 * 1}{8}\right)} * 0.60$$

$$N = 40{,}132.4399 * 0.60$$

$$N = 24{,}079.464$$

TMG is the total memory available in gigabytes, NN is the number of nodes and PM is the percentage of the memory to use. According the physical characteristics of our test server we choose a

60% use of memory RAM.

To obtain an exact result for N, an additional operation with the NB variable needs to be done. In this case 96 was used for the NB value which is within the permitted values of HPL.

1) $N = 24{,}079.464/96$ 3) $N = 250 * 96$

2) $N = 250.827$ 4) $\mathbf{N = 24{,}000}$

Similar to the Top500 HPL Calculator (Sindi, 2016) is in the third step of the calculation only the integer part of the division of N/NB taken. This result needs to be multiplied again with NB to obtain the value N for the size of the problem to be solved.

3. RESULTS AND DISCUSSIONS

3.1. CPU performance tests

The testing environment was configured with the aid of the Top500 HPL Calculator (Sindi, 2016) as to optimize the performance and maximize the use of the available resources. The size of the problem, N, was determined 24,000 and the block size NB 96. The test was performed with 4, 8 and 16 instances. Figure 2 shows that the Docker performs in all three scenarios around one Gflop better than the configuration with the VM.

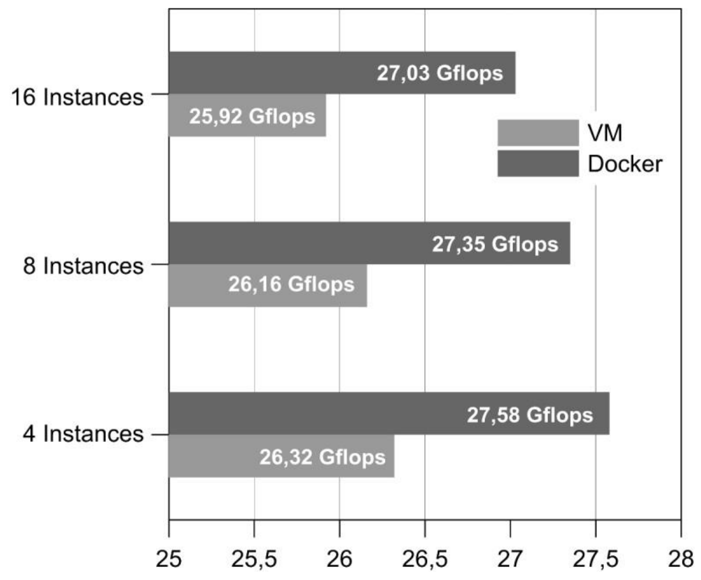

Figure 2. Test results of the Gflops CPU performance.

3.2. RAM performance tests

Sysbench is a multithreaded tool that allows performance testing under heavy load: RAM memory, CPU, memory I/O, disk I/O and MySQL database. In our case, it was used for testing the RAM memory for which the tool was installed in the VM and in the Docker container. Both environments were configured with the following parameters:

1. memory-block-size = 12GB
2. memory-total-size = 8GB

The Docker container performs 137 MB/s, is faster than the VM which represents a 4.5% of better uses of its resources. Figure 3 depicts the data transfer rate for Docker and VM, respectively.

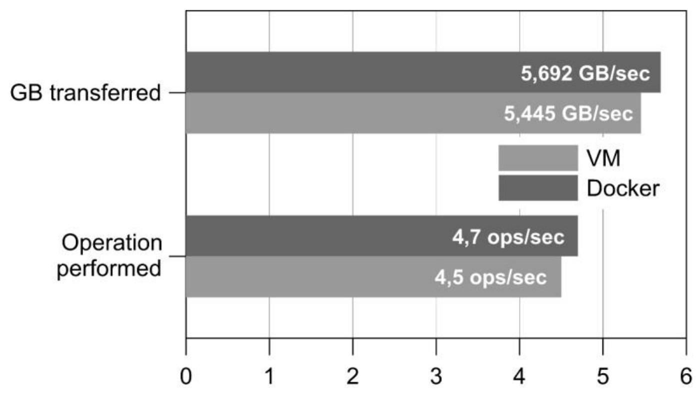

Figure 3. Results of the RAM performance tests.

3.3. Network performance tests

Iperf allows performance testing of the computer networks by creating flows of TCP and UDP data. The tests with iperf were conducted within a range of 60 seconds in client and server mode. The results obtained are shown in the Figs. 4 and 5.

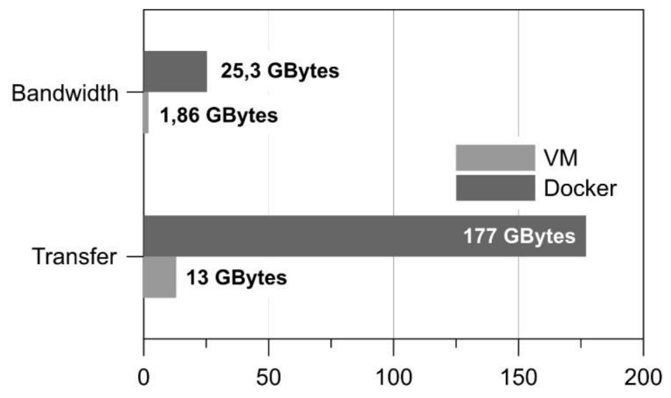

Figure 4. Results of the network Server Transfer performance.

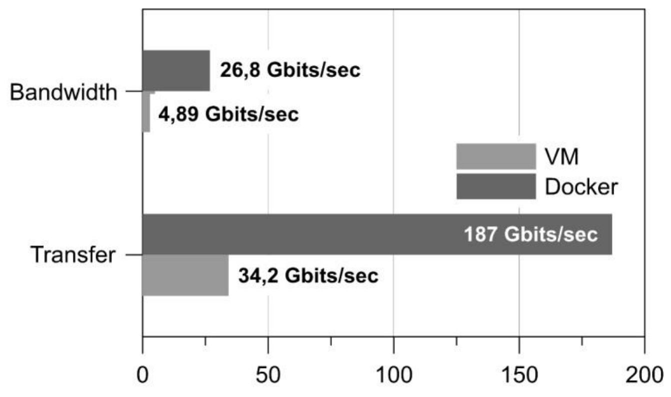

Figure 5. Results of the network Client Transfer performance.

The network performance is 12.6 times higher in the container on client mode; in server mode is the container 4.5 times faster than the VM. The relative large differences are due to the fact that the hypervisor of the VM creates a default limited virtual network adapter, during high data transmission. The virtual network adapter is a restriction for the traffic causing a bottleneck, slowing down the communications. Docker also creates on the host a virtual interface, named docker0. That interface uses all available velocity. When a Container is created, the pair of peer interfaces connect each of the

Containers to the "docker0" bridge and automatically forwards packets between any other network interface that is attached to it.

3.4. Disk performance tests

Bonnie++ is a lightweight C++ tool for testing and measuring the performance of hard drives. In our case the test was performed with an 8GB data file. Findings are revealed in Fig. 6. In Docker we used the default open source storage driver AUFS and in VirtualBox the default disk driver SATA.

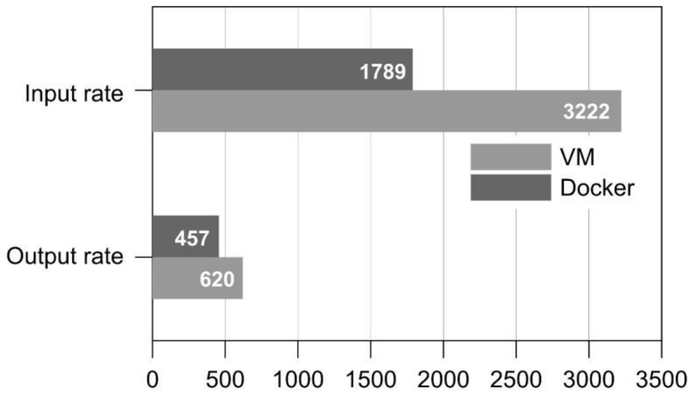

Figure 6. Results of the disk performance test.

The results of the hard disk test show that in the given test case the container has a lower yield compared to the test results of the VM (Wes Felter, 2014), and that AUFS introduce significant overhead in I/O because the data is going through several layers. The data container is 26.3% slower in reading and 44.5% in writing data. In our test setup is VM clearly faster in file handling.

4. CONCLUSIONS

The container technology is growing fast and offers several characteristics that are worth evaluating. In this research, we focused on performance assessment. According to the obtained results offer Docker containers more advantages than VirtualBox from an efficiency point of view. The container solution uses more efficiently the available hardware resources, especially form the network perspective. However, the read and writing performance is an area where the VM is outperforming the container solution.

We can conclude that using containers for Software Applications with little I/O perform better on a container based infrastructure and therefore do need less hardware resources. In further studies, it could be of interest to test different storage drivers for the container environment such as btrfs, device mapper or vfs to determine differences in performance and architecture.

BIBLIOGRAPHY

Petitet, A. P., Whaley, R. C., Dongarra, J., & Cleary, A. (2008). *HPL - A portable implementation of the high-performance linpack benchmark for distributed-memory computers*. Available at https://www.researchgate.net/publication/244449955_HPL_-_a_Portable_Implementation_of_the_High-Performance_Linpack_Benchmark_for_Distributed-Memory_Computers

Barik, R. K., Lenka, R. K., Rao, K. R., & Ghos, D. (2016). *Performance analysis of virtual machines and containers in cloud computing*. 2016 International Conference on Computing, Communication and Automation (ICCCA), 1204-1210. doi:10.1109/CCAA.2016.7813925

Jacobson, D. M., & Canon, R. S. (2015). *Contain this, unleashing Docker for HPC.* Available at https://www.nersc.gov/assets/Uploads/cug2015udi.pdf

Chung, M. T., Quang-Hung, N., Nguyen, M. T., & Thoai, N. (2016). *Using Docker in high performance computing applications*. 2016 IEEE Sixth International Conference on Communications and Electronics (ICCE), 52-57. doi:10.1109/CCE.2016.756261, R. (n.d.). *Bonnie++*. Retrieved from http://www.coker.com.au/bonnie++/

Spoiala, C. C., Calinciuc, A., & Turcu, C. O. (2016). *Performance comparison of a WebRTC server on Docker versus Virtual Machine.* Retrieved 08 10, 2016, from 13 th International Conference on Development and Application Systems: http://www.dasconference.ro/dvd2016/data/papers/D78-paper.pdf

Docker. (2017). *Docker*. Retrieved from What is Docker: https://www.docker.com/what-docker

Kleyman, B. (2012). *Hypervisor 101: Understanding the virtualization market*. Retrieved from http://www.datacenterknowledge.com/archives/2012/08/01/hypervisor-101-a-look-hypervisor-market/

Morabito, R. (2016). *A performance evaluation of container technologies on Internet of things devices*. 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 999-1000. doi:10.1109/INFCOMW.2016.7562228

Parziale, L., Louie, B., Marins, E., Santos, T. N., & Venkatesan, S. (2012). *Advanced networking concepts applied using Linux on IBM Systems*. Retrieved from Performance tests and results: http://www.redbooks.ibm.com/redbooks/pdfs/sg247995.pdf

Peter Mell, T. G. (2011). *The NIST definition of cloud computing.* Retrieved from http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf

Pipes, J. (2010). *Performance tuning best practices.* Retrieved from http://docs.linuxtone.org/ebooks/MySQL/performance-tuning-best-practices.pdf

Preeth, N. E., Mulerickal, F. J., Paul, B., & Sastri, Y. (2015). *Evaluation of Docker containers based on hardware utilization.* 2015 International Conference on Control Communication & Computing India (ICCC), 697-700. doi:10.1109/ICCC.2015.7432984

Sindi, M. (2009). *HPL Calculator.* Retrieved from http://hpl-calculator.sourceforge.net/HPL-HowTo.pdf

Sindi, M. (2016). *Top500 HPL calculator*. Retrieved from http://hpl-calculator.sourceforge.net/

Wes Felter, A. F. (2014). *An updated performance comparison virtual machines and Linux containers.* Retrieved 08 06, 2016, from IBM Research Report : http://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/$File/rc25482.pdf

Xavier, B., Ferreto, T., & Jersak, L. (2016). *Time provisioning evaluation of KVM, Docker and Unikernels in a cloud platform.* 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 277-280. doi:10.1109/CCGrid.2016.86