High-Performance Computing as a Cloud Computing Service:

Virtualization or Containerization?

Safae Bourhnane
School of Science and Engineering
Al Akhawayn University
Ifrane, Morocco
s.bourhnane@aui.ma

Mohamed Riduan Abid
School of Science and Engineering
Al Akhawayn University
Ifrane, Morocco
r.abid@aui.ma

Abstract—High-Performance Computing (HPC) has emerged as an appealing Cloud Computing (CC) service, especially with the proliferation of big data. Virtualization is the CC main technology enabler. Nevertheless, due to its light-weight, containerization is rising as a promising alternative. In literature, there is considerable work comparing the performance of virtualization and containerization. However, to our knowledge, none is comparing this performance from an HPC as a Service (HPCaaS) perspective. In this paper, further light is shed on the advantages and disadvantages of using Virtual Machines (VMs) and containers. We study their performance from an HPCaaS perspective by deploying a real-world private Cloud and running relevant HPC jobs using real-world big data samples.

Keywords—Virtualization; Cloud Computing; HPC; Containerization; HPC; Virtual Machines;

I. INTRODUCTION

With the recent emergence of pervasive computing [1], whereby network connectivity is everywhere and uses many devices, extravagant amounts of data are generated, which fall within the realm of big data. Dealing with big data brings two main challenges: storage and processing.

To process big data, HPC is necessary. There are only two solutions to deploy HPC: 1. using supercomputers or 2. building clusters of computers. The second solution is the most promising one as it is cost-effective. Because of the continuous increase in the ratio of computers performance to their price, building clusters has became the HPC trend. Nevertheless, today's companies are seeking more profit and looking for solutions to reduce the overhead introduced by building clusters of physical machines (cost of the physical machines, maintenance, electricity, the staff, etc.). In this context, the Cloud emerges as the most cost-effective solution.

Cloud Computing (CC) is of great interest to a variety of companies due to the new paradigm it brings. CC allows for computing to be treated as a utility. This implies that needed resources will always be available, ready-to-use, and users will be only charged for what they consumed (i.e., pay-per-use), exactly like any other utility, e.g., gas, water, telephony.

When delving into the fundamentals of Cloud Computing operation, it became clear that virtualization is a key technology enabler. Indeed, Cloud services are allocated via the instantiation of machines (VMs): VMs are "forked" at the beginning of the CC service request and are shut down at release time, a fact that permits a real-time metering of the usage and thus easing pay-per-use deployment.

Regarding virtualization, VMs are the norm. However, recently, containers started emerging as a promising alternative to VMs because of the advantages they provide, mainly being lightweight, compact (i.e., self-contained), and their loose-coupling with the underlying operating system. These advantages are discussed later in the paper.

In literature, considerable work has been carried out about comparing VMs and containers [2] [3] [4]. However, no literature work exists on comparing them from an HPC as a Service (HPCaaS) perspective. While reviewing the work previously done, it became clear that the majority of the previous works compare the architecture and other characteristics with a noticeable absence of HPCaaS.

In this paper, an evaluation and comparison of the performance of VMs and containers from an HPCaaS perspective are conducted in order to study the possibility of using containers for HPC jobs. To this end, a real-world private Cloud has been deployed and both clusters of VMs and containers have been implemented. Open-source platforms were used, mainly Docker [5], Hadoop [6], and Openstack [7]. Besides, the study is based on the TeraSort benchmark [8] to run HPC jobs and measure the response time of each cluster in seconds.

The rest of the paper is organized as follows. In Section II, relevant background is presented. Section III delineates the differences between VMs and containers. In Section IV, highlights about the fundamentals of HPCaaS are presented. Section V presents experimental settings, results, and discussion. Finally, the conclusion is presented in Section VI.

II. BACKGROUND

National Institute of Standards and Technology (NIST) defines Cloud Computing as a way to provide ubiquitous and on-demand network access to a shared pool of resources [9]. This allows rapid resources provisioning and release with little effort and with almost no interaction with the provider.

In 1961, John McCarthy was the first one to introduce the idea of computing as a utility [10]. Utility conveys two main characteristics: availability and pay-per-use. This stipulate means that resources are accessed whenever needed, and users pay only for what has been consumed.

Behind Cloud Computing lies virtualization as the main technology enabler. More precisely, the pay-per-use feature is implemented via tracking the start time and the end time when the VM was allocated and released for a particular Cloud service, and by considering the size of the VM.

For instance, in the well-known Openstack platform, different VM sizes are already set by default and are called "flavours". One can choose the VM size/flavour that best matches the requested CC service demand or even create a customized flavour. The VM sizes/flavours are mainly defined by the RAM memory size, disk, number of vCPUs, and RxTx factor (i.e., network bandwidth).

According to NIST [11], a virtualized environment simulates the software or the hardware on which other applications are running. This virtual environment is called *virtual machine*. Nowadays, virtualization can be associated to many other resources (storage, network, etc.). The virtual instances of these resources are provided via the *hypervisor*. A hypervisor is a thin layer of software that runs either on top of the hardware (type-1 hypervisor), or on top of the OS (type-2 hypervisor) [12].

Recently, containerization started emerging as a promising alternative to virtualization as it mainly leverages a lightweight implementation of the virtualization principles [13], e.g., consolidation and isolation.

Containerization involves encapsulating the application with its libraries and other run-time dependencies/components. Unlike virtualization, it is an OS-level virtualization technique that helps launching applications without instantiating an entire VM, and thus without involving a hypervisor. There can be multiple applications running on the same host and sharing the same OS kernel. These applications are completely isolated from the user's perspective.

Applications' containers consume fewer resources compared to VMs because they share the same resources without the need for a full dedicated OS. An image is all that is needed to run a container. These images can be either built by the user or downloaded from registries. Containers are meant to operate independently from each other, and each container communicates with others through an API (Application Programming Interface).

There are several containerization technologies around, however, Docker is still the most appealed [14]. It is widely used because it is an open-source solution that is easy to deploy. Also, it has a large community that has been contributing to its development over the past years and providing continuous maintenance and updates.

Though containerization exhibits tangible advantages over virtualization, both technologies are still strongly-coupled to the type of applications and thus heavily dependent on the application scenario, i.e., they are application-specific.

For instance, in Mobile Cloud Computing [15], whereby mobile users are continuously migrating from a Cloudlet [16] to another, containerization proves the best as it leverages lightweight allocation of resources. On the other hand, for applications that are long-lasting, e.g., Web servers and database application, virtualization proves the best. In this paper, further light is shed on this trade-off and more focus is put on the appropriateness of the two technologies from an HPC as a Cloud service perspective.

III. VIRTUALIZATION VS. CONTAINERIZATION

A. Virtual Machines

VMs are nothing but a software construct that mimics the features of the physical server [17]. It has a limited number of vCPUs (virtual CPU) and a limited amount of memory. Once forked, the VM starts acting just like a physical machine simply because the users can still use the same interaction interface (e.g., APIs) as with physical machines. However, the VM is only aware of the resources that have been allocated to it and does not see the resources of the host. This is due to the fact that the installed OS (in the VM) still "thinks" that he owns the hardware resources: The VM OS still sends binary instructions at the ISA (Instructions Set Architecture) Interface. Still, instead of instructions being sent to the hardware, they are caught by the real OS of the machine, i.e., the hypervisor, and then they are either translated or modified depending on the adopted virtualization technology, e.g., full or para-virtualization, and type-1 or type-2.

a. Advantages of Virtual Machines

One of the main advantages of VMs is consolidation [18] whereby a single physical server can host multiple (virtual) machines. This has a direct, and tangible, impact on the reduction of the number of physical servers in a data center, the electricity bill, and for the maintenance and management labor. Another benefit is the ability to dynamically provision needed resources, a.k.a. elasticity. This allows companies and organizations to respond to their needs in terms of resources in a matter of minutes or even seconds. Companies are no longer forced to own physical servers to respond to the peak demand they sporadically witness, e.g., during Christmas holidays. With the advantage of VM migration, virtualization can cope with stringent, and sudden, congestion in VMs requests. For instance, in Mobile Cloud Computing, whereby users are constantly moving, assigned VMs can move (i.e., migrate) with the users in order to maintain an optimal latency time. Besides, for Cloud centers witnessing congestion (i.e., higher loads), VMs can be migrated to other center in order to balance the load.

Last, but not least, the fundamental advantage of virtualization is providing isolation. This means that VMs are behaving exactly like real-world physical machines when it comes to security. In other words, even if the physical machine hosting the VMs is sharing its resources (memory space, IO, and CPU) among the VMs, appropriate mechanisms are implemented (e.g., vCPUs, virtual paging) in such a way that each VM's resource is completely isolated from others. For instance, if a VM gets

contaminated with a virus, other VMs in the same physical machines are still intact.

b. Disadvantages of Virtual Machines

The main disadvantage of VMs is their heaviness in terms of needed resources. VMs consume more resources basically because of the need to "fork" an appropriate OS (Operating System) for each VM, and thus inducing longer time for starting it up. Besides, and regarding migration, VMs take significant time to move from a station to another.

The constraint on the heavy load for VMs limits the capacity of a physical machine in terms of the maximum number of VMs to host. Thus, having multiple VMs running on the same physical host at the same time may introduce an unstable performance of the system due to the workload brought by each of the VMs running. Consequently, and even if users can perform the exact same operations on a VM (i.e., using the same interface) as on a physical host, they should not expect to have the same performance as the one of the physical host.

B. Containers

Containers are isolated environments where you can deploy your application along with all its dependencies and libraries. They have some mutual features with VMs, but they do not share most of their properties. Just like VMs, containers have access to the host resources (compute, storage, and network). However, and unlike VMs, containers share the same OS kernel, but they keep the applications isolated from each other. In other words, unlike a VM, a container does not have its own OS, and thus there is no need to deploy a hypervisor. The latter is replaced by appropriate containerization software, e.g., Docker.

Figure 1 shows the difference between containers and virtual machines.

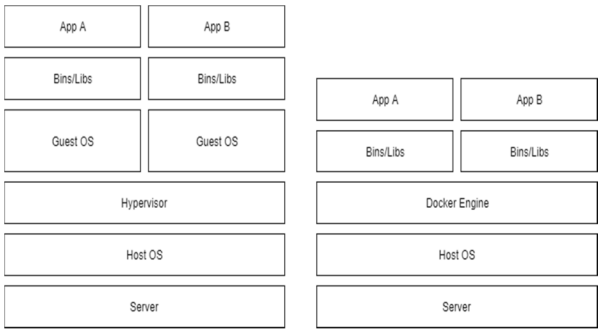

Figure 1. VMs Vs. Containers

a. Advantages of Containers

Unlike VMs, containers are lighter in terms of resources need. A typical container size is in the order of tens of MBs. Thanks to their lightness, containers resources provisioning takes few seconds, thus decreasing the start time, and increase user satisfaction via the decrease of the response time.

Containers are cost-effective and present a promising solution for micro-services and continuous deployment, however, they are not suitable for demanding and heavy applications.

b. Disadvantages of Containers

As a main disadvantage, Containers are not completely isolated as they share the kernel and other components of the host, and this poses significant security threats. Besides, containers are strongly-coupled to the underlying OS, i.e., if a container is launched in a Windows environment, it cannot be easily migrated and deployed into a Linux environment. This will require specific manipulation. Last, but not least, networking between containers that are completely isolated (running on different hosts) can be tricky and not easy to achieve. It takes a specific configuration of the network which involves adding an additional virtual bridge.

IV. HPC AS A CLOUD SERVICE

With the emergence of pervasive and ubiquitous computing [19], the world of things (a.k.a Internet of Things) [20] keeps continuously generating huge amounts of data, i.e., big data. The latter needs two main resources: storage and processing.

Big data needs to be stored appropriately to ease the process of retrieval and processing later. This storage usually has to do with the type of applications that these data is to be used for. Besides, the adopted storage solution needs to protect the data from all types of frauds.

Regarding the processing, it goes without saying that Big data requires HPC (High Performance Computing). This latter can be offered using only two solutions: either by owning supercomputers or building clusters of commodity hardware. The last solution is the most adopted thanks to its cost-effectiveness. However, there is always need for better solutions. HPC mainly applies the concept of supercomputers to solve problems related to computations that are too large or time consuming for commodity hardware. An HPC system has thousands of processors, a large memory and a huge amount of storage.

Companies have noticed that the physical cluster they build, to provide HPC, requires many resources that they could not constantly afford. Better, the cost of buying the hardware itself, in addition to the cost of electricity, maintenance, working labor and others, can be avoided. All these expenses can be avoided if the HPC is to be provided by the Cloud, i.e., using HPCaaS Cloud service (HPC as a Service).

The Cloud allows companies to benefit from many resources, namely computing resources, without owning physical servers. Furthermore, the Cloud allows for the pay-per-use concept which saves cost to companies by having them only pay for what they have consumed.

As mentioned previously, virtualization is the technology enabler number one behind Cloud computing, that is because Cloud services are offered via the instantiation of VMs. However, containers are emerging as a promising alternative to virtualize the resources, and that is gaining a lot in terms of popularity. With regards to the advantages and disadvantages of VMs and containers, further investigations are led to determine which one is optimal for HPC over the cloud.

V. Experimentation and Discussion

The experiments led are based on a popular data set benchmark that is called TeraSort. It is used to measure the performance of MapReduce through sorting 1 TB of randomly distributed data over a computer system.

To compare VMs and containers for HPCaaS, two main sets of experiments were conducted:

- Set #1: compares a cluster of distributed VMs and a cluster of distributed containers. Each VM, and container, is deployed in a single physical machine.
- Set #2: compares a cluster of centralized VMs and a cluster of centralized containers and a cluster of centralized containers with limited resources. The VMs set, and the containers set, lie in the same physical machine.

Openstack was used to create the private cloud and launch the VMs. Docker was used to run the containers, and to assess the performance Hadoop was used with the TeraSort benchmark.

The hardware used for the experiment setup is Dell OptiPlex 390 with the following specifications:

TABLE I DELL OPTIPLEX 390 TECHNICAL SPECIFICATIONS

Dell OptiPlex 390 Technical Specifications	
Processor	Intel® Core™ i5 Quad Core; Intel® Core™ i3 Dual Core; Intel® Pentium® Dual Core; Intel® Celeron® Dual Core
Memory	Up to two DIMM slots; Non-ECC dual-channel 1333MHz DDR3 SDRAM, up to 8GB
Operating System	Ubuntu 16.04 LTS
Hard Disk Drive	3.5” Hard Drives: up to 1TB 7200 RPM SATA 3.0Gb/s Supports Dell’s Flexible Computing Solution diskless option

A. *Cluster of distributed VMs Vs. Cluster of distributed containers.*

a. *Experiment Setup*

The general architecture of the cluster of the VMs is shown in Figure 2.

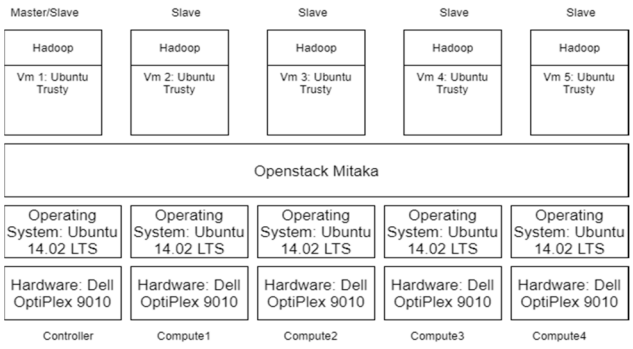

Figure 2. Cluster of Distributed VMs

The bottom layer contains the physical servers that were used to install Openstack. Then comes the OS layer that is directly under Openstack Mitaka. On top of it, there is a layer of five virtual machines each one consisting mainly of an operating system (Ubuntu trusty), and the framework Hadoop.

Regarding the cluster of distributed containers, the architecture is shown in Figure 3:

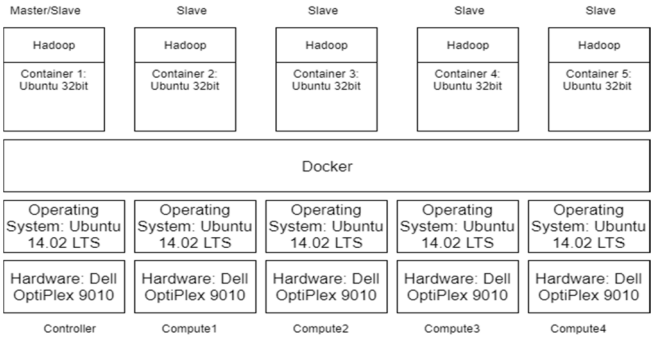

Figure 3. Cluster of Distributed Containers

The bottom two layers are the same as the previous architecture. However, instead of having Openstack in the third layer, there is Docker. Then five containers that are running an Ubuntu 32bit image with Hadoop installed on top were launched.

b. *Results and Comparison*

The graph in Figure 4 shows the comparative average time for Terasort, with different data set sizes, between the two clusters. It uses four different data set sizes and shows the average running time (in seconds) of the Terasort.

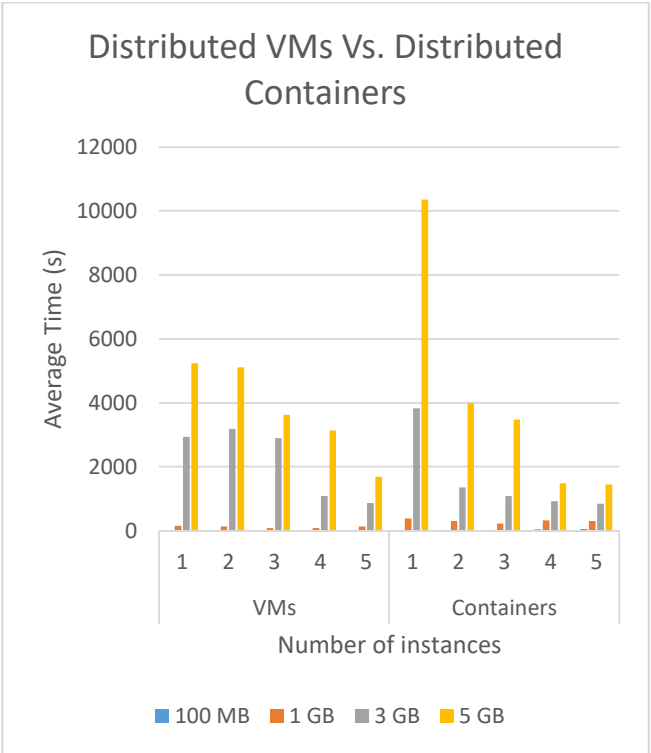

Figure 4. Distributed VMs Vs. Distributed Containers

Table II describes the difference in the performance (the average running time) between the distributed VMs and the distributed containers. It presents the increase or decrease of the performance when going from using VMs to using containers for all the datasets used and for the different numbers of nodes in the clusters. The decrease of performance is represented by (-) and the increase is represented by (+).

TABLE II DISTRIBUTED VMS VS. DISTRIBUTED CONTAINERS

Distributed VMs Vs. Distributed Containers				
	100MB (%)	1GB (%)	3GB (%)	5GB (%)
1	-28.57	-61.24	-23.20	-49.4
2	-30	-55.5	+37.99	+21.9
3	-52.2	-61.26	+42	+6.9
4	-59.35	-72.7	+15.05	+52
5	-54	-56	+1.85	+14.5

As it can be inferred from the Table II, when going from VMs to containers a loss in performance was noticed when dealing with datasets of 100MB and 1GB (loss of up to 72% of the performance). However, when increasing the size of the dataset used to 3GB and 5GB, a noticable increase in the performance that reached 52% was found.

B. Cluster of centralized VMs Vs. cluster of centralized containers Vs. cluster of centralized containers with limited resources

a. Experiment Setup

The architecture of the cluster of centralized VMs is shown in the following figure (Figure 5).

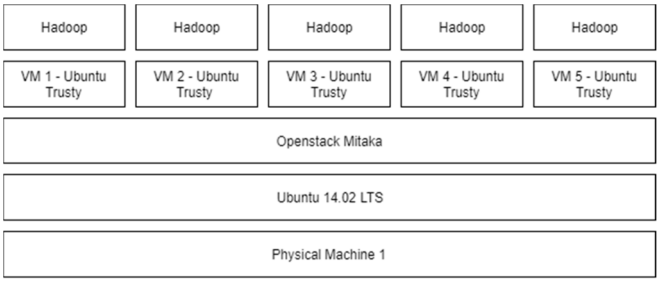

Figure 5. Cluster of Centralized VMs

Unlike the previous architecture, in this one all the VMs are deployed in one physical machine.

The other two clusters (centralized containers and centralized containers with limited resources) have the same architecture that is shown in the Figure 6.

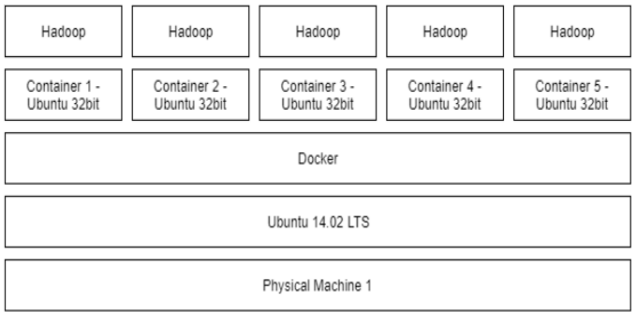

Figure 6. Cluster of Centralized Containers

In the cluster of centralized containers with limited resources, the containers are given the same resources as the VMs in order to have a better comparison. As the VMs used are of the *small* flavour (1VCPU and 20GB Disk), the resources of the containers were limited using the following command: *docker run ubuntu -h master -it -cpus=1 -m=2000000*".

The *'-cpus'* means that the number of CPUs to be used by this container is limited to one, and the container is given 20GB of memory that it can use. If the container needs more than the resources allocated to it, there is no way it can take more from the host because the latter does not support the wrapping.

b. Results and Comparison

The graph in Figure 7 shows the results of running TeraSort on the three different clusters.

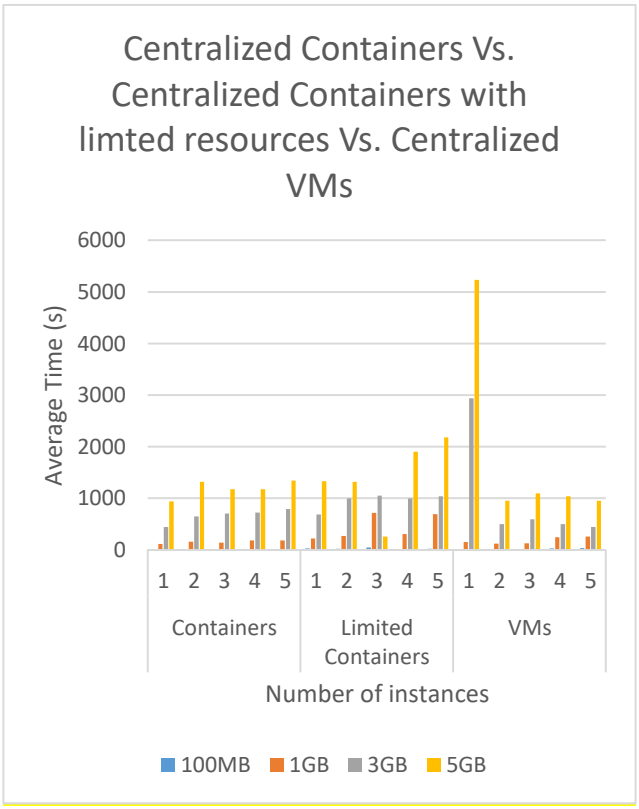

Figure 7. Centralized Containers Vs. Centralized Containers with limited resources Vs. Centralized VMs

The following table (Table III) shows the percentage of the decrease or increase that happened to the performance when going from a cluster of centralized VMs to a cluster of centralized containers with unlimited resources. The decrease of performance is represented by (-) and the increase is represented by (+).

From Table III, it is clear that the cluster of centralized containers performs better when there is only one node in the cluster and with the smallest dataset (with 100MB up to 57.57% gained of the running time, while there was a loss of up to 78.73% of the performance when running a cluster of 5 containers and when running a dataset of 3GB).

TABLE III CENTRALIZED VMS Vs. CENTRALIZED CONTZINERS WITH UNLIMITED RESOURCES

Centralized VMs Vs. Centralized Containers with unlimited resources				
	100MB (%)	1GB (%)	3GB (%)	5GB (%)
1	13.33	22.66	84.77	82.01
2	-25	-29.03	-28.62	-38.30
3	0	-10.31	-18.38	-7.19
4	51.85	24.79	-45.38	-13.21
5	57.57	29.72	-78.73	-41.26

The difference of performance between the virtual machines and the containers with the limited resources through is presented in the following table (Table IV). The decrease of performance is represented by (-) and the increase is represented by (+).

TABLE IV CENTALIZED VMS VS. CENTRALIZED CONTAINERS WITH LIMITED RESOURCES

Centralized VMs Vs. Centralized Containers with limited resources				
	100MB (%)	1GB (%)	3GB (%)	5GB (%)
1	-93.33	-46	76.63	74.56
2	-75	-120	-97.81	-38.19
3	-92.30	-150	-77.40	-27.50
4	33.33	-25.60	-99.79	-83.70
5	36.36	-168.33	-135.29	-129.57

C. Discussion

This section summarizes the findings and explains the results of the comparison done in the previous sections.

As noticed from previous results, containers are better than virtual machines when it comes to performing small tasks. Also, as more containers join the cluster, the performance decreases significantly. As it might be clear, HPC needs many instances and cannot be done using only few nodes in the cluster.

For containers to be connected to each other in a distributed system there is a need to add an additional layer of virtual network. In this setup, the OpenVSwitch was used, which acts like an advanced edge between the containers allowing them to ping and *ssh* each other. Adding this layer had an impact on the overall performance of the cluster and made the network overhead noticeable especially with small data sets. This overhead is not tolerable when dealing with HPC jobs because HPC applications are already expensive and cannot support any additional costs.

Containers perform better than virtual machines as single nodes because there need no hypervisor and run directly on the operating system just like regular processes.

Containers suffer from a problem that is the slow IO performance compared to VMs. The results in [21] show that the performance of the containers when it comes to IO operations is about 40% less than the host. In [22], containers performance issues were further assessed using MySQL databases and some OLTP operations. They found out that containers perform a little less than virtual machines.

Morabito in [23] gives some tips for enhancing the performance of the containers. It is mentioned that containers are best at running applications that can be divided into multiple micro-services. Besides, to have the best out of the containers, they have to be dealt with as a type of process virtualization rather than machine virtualization.

In addition to that, an experiment has been done to compare Kernel-based Virtual Machine (KVM) and the bare metal [24]. It has shown that KVM delivers near bare metal performance. That is mainly because virtual machines are older (1960s) and hence more mature than containers (1980s). This maturity helped improving the performance as it developed a larger community that has been exploring it and solving many problems related to it.

It is important to mention that containers are lighter and have a smaller starting time compared to the virtual machines. Figure 8 shows the boot and reboot time of both containers and virtual machines.

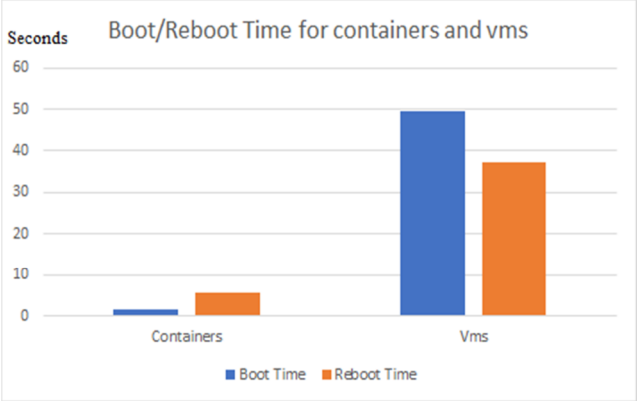

Figure 8. Boot/Reboot Time for Containers and VMs

Containers boot faster because their image does not include the operating system, unlike the virtual machine. Besides, to launch an instance using Openstack there is a heavy process that involves many components of the Openstack platform (networking, scheduling, and others).

Figure 9 shows the difference in size of the container image and the virtual machine image. Both images have Hadoop installed.

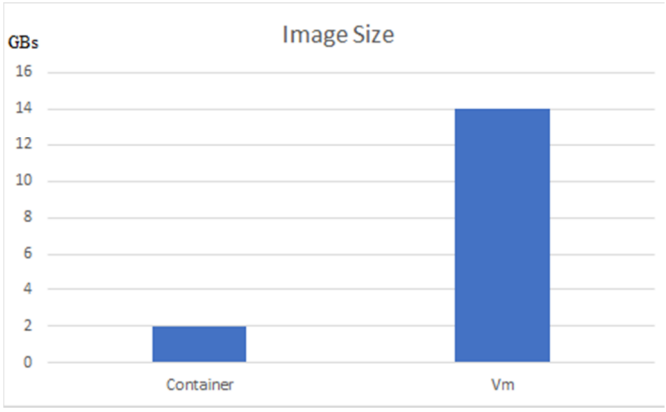

Figure 9. VM image size Vs. Containers image size (GBs)

Figure 9 shows that a VM image has about 14 Gigabytes (GBs) with hadoop installed while a Docker container has only 2 GBs.

VI. Conclusion and Future Work

This paper aimed at showing the difference between running Hadoop on a cluster of virtual machines and a cluster of Docker containers. All this was for the sake of determining the best virtualization technology to use from a HPCaaS perspective.

For this, two different sets of experiments were conducted. The first set of experiments consisted of comparing a cluster of distributed virtual machines and a cluster of distributed containers. The second set was about comparing a cluster of centralized virtual machines (forked in the same compute node), a cluster of centralized containers (launched on the same physical machine), and a cluster of centralized containers with limited resources. Each of the clusters was tested using the TeraSort benchmark that was performed using four different datasets (100MB, 1GB, 3GB, and 5GB).

The findings of this study showed that containers do not provide the required performance of the HPC applications. It was observed that the network overhead introduced in the cluster of containers is more significant. Besides, it became clear that containers perform better with small datasets (eventually small tasks), and since HPC is about dealing with big tasks and large datasets, containers do not seem like the solution to be adopted. Furthermore, the findings of this study sustain what has been mentioned in the literature and that claimed that containers also have performance issues. Thus, the best solution for HPC remains virtual machines.

As future work, an extension of this study is expected by trying the different clusters with bigger sizes of datasets. Also, the scalability of the clusters will be tested by adding more nodes. Furthermore, the eventual studies will be based on the real-world data generated from a real deployment of a Wireless Sensor Network and then process it using one of the virtualization technologies studied in this paper.

References

[1] Technopedia Inc. , «Pervasive Computing,» 2017. [Online]. Available: https://www.techopedia.com/definition/667/pervasive-computing. [retrieved November 2018]

[2] L. Chaufournier. P. Shenoy Y., and Tay. Prateek Sharma, «Containers and Virtual Machines at Scale: A Comparative Study,» Middleware' 16, DOI: http://dx.doi.org/10.1145/2988336.2988337, 2016.

[3] R. Dhar, «Comparative Evaluation of Virtual Environments: Virtual Machines and Containers,» University of Dublin, Trinity College, 2016.

[4] M. Eder, «Hypervisor vs. Container-based Virtualization,» Network Architectures and Services, doi: 10.2313/NET-2016-07-1_01, 2016.

[5] «Why Docker?,» [Online]. Available: https://www.docker.com/why-docker. [retrieved November 2018].

[6] «Apache Hadoop,» [Online]. Available: https://hadoop.apache.org/. [retrieved November 2018].

[7] «Openstack,» [Online]. [retrieved November 2018].

[8] «Terasort Benchamrk,» [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SSZUMP_7.2.1/mapreduce_integration/map_reduce_terasort_example.html. [retrieved November 2018].

[9] P. Mell and T. Grance, The NIST Definition of Cloud Computing, Gaithersburg: National Institute of Standards and Technology, 2011.

[10] I. Chana and T. Kaur, «Delivering IT as a Utility - A Systematic Review,» IJFCST, vol. 3, n° 13, pp. 11-30, 2013.

[11] M. Souppaya and H. Karen Scarfone, Guide to Security for Full Virtualization Technologies, Gaithersburg: National Institute of Standards and Technology, 2011.

[12] R. P. Goldberg, «Architecture of Virtual Machines,» AFIPS National Computer Conference, pp. 74-112, 1973.

[13] M. Eder, «Hypervisor- Vs. Container-based Virtualization,» Seminars FI / IITM WS 15/16, pp. 1-7, 2016.

[14] J. Turnbull, The Docker Book, 2015.

[15] A. Sarker. A. Newaz. Bahar. M. Atiqur. Rahman., and S M Shamim, «A Review on Mobile Cloud Computing,» International Journal of Computer Applications, vol. 113, n° 116, pp. 4-11, 2015.

[16] F. V. Vargas, «Cloudlet for the Internet of Things,» KTH, Stockholm, 2016.

[17] R. N. James and E. Smith, «The Architecture of Virtual Machines,» IEEE Computer Society, pp. 32-40, 2005.

[18] M. Najafzadeh. M. Sharifi., and H. Salimi, «Advantages, Challenges and Optimization of Virtual Machine Scheduling in Cloud Computing Environments,» International Journal of Computer Theory and Engineering, vol. 4, n° 12, pp. 189-193, 2012.

[19] L. Bhasker. T, «Pervasive Computing Issues, Challenges and Applications,» International Journal of Engineering and Computer Science, vol. 2, n° 112, pp. 3337-3339, 2013.

[20] E. Sayed. Ali. Ahmed. Z. , and Kamal Aldein Mohammed,, «Internet of Things Applications, Challenges and Related Future Technologies,» World Scientific News, vol. 67, n° 12, pp. 126-148, 2017.

[21] Alkmim, «Slow IO Performance Inside Container Compared With the Host,» 2016. [Online]. Available: https://github.com/moby/moby/issues/21485. [retrieved November 2018]

[22] A. Ferreira. R. Rajamony. J. Rubio., and W. Felter, «An Updated Perfomance Comparison of Virtual Machines and Linux Containers,» Performance Analysis of Systems and Software, 29-31 March 2015.

[23] R. Morabito, «A Performance Evaluation of Container Technologies on Internet of Things Devices,» IEEE Infocom, pp. 1-2, 2016.

[24] GrayBoltWolf, «How Fast is KVM? Host Vs. Virtual Machine Performance!,» [Online]. Available: https://forum.level1techs.com/t/how-fast-is-kvm-host-vs-virtual-machine-performance/110192. [retrieved November 2018]