Impact of Hybrid Virtualization Using VM and Container on Live Migration and Cloud Performance

Chapter · January 2019

DOI: 10.1007/978-3-319-91337-7_19

2 authors:

Oussama Smimite
University Ibn Zohr - Agadir
5 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Afdel Karim
University Ibn Zohr - Agadir
124 PUBLICATIONS **528** CITATIONS
SEE PROFILE

Impact of Hybrid Virtualization Using VM and Container on Live Migration and Cloud Performance

Oussama Smimite(✉) and Karim Afdel

Laboratory Computing Systems and Vision - LabSiV, Ibn Zohr University, Agadir, Morocco
oussama.smimite@edu.uiz.ac.ma, k.afdel@uiz.ac.ma

Abstract. Hypervisor-based virtualization and containerization offer abstraction capabilities that make applications independent of the cloud's hardware infrastructure for migration. Live migration is a widely used technology for load balancing, fault tolerance and energy saving in Cloud Datacenters. In this paper, we will be converging the two types of virtualization (Container and VMs together) into a hybrid architecture that promotes their complementarities without altering or degrading the quality of service (QoS) of the Cloud infrastructure. For this, we studied the live migration of a monolithic application and precisely the video streaming installed in a container nested in a virtual machine. We examined the contribution of this new type of virtualization in terms of processor utilization, disk usage, total migration time and downtime. The results presented show the advantages of Hybrid virtualization especially in terms of total migration time. Indeed, the absence of a real virtualization layer in the containers leads to a rapid interaction with the operating system processes, which affects the total migration time.

Keywords: Cloud · Virtualization · Live migration · VM container
Hybrid virtualization · Performance migration · Availability

1 Introduction

Cloud computing has multiple advantages for Software as a Service: portability, upgradability, scalability, high availability, cost efficiency, and resource sharing [1]. For these reasons, operators and software providers are switching to cloud computing for their infrastructure to gain more flexibility and optimize their costs.

For the cloud data centers, virtualization technologies as Xen [2], Hyper V [3] and VMware [4] are widely used for their ease of use, cost efficiency, flexibility of resource management, scalability and the simplicity of enabling the high availability.

More precisely, virtualization offers the possibility of a fine-tuning of resources allocation by associating processors, RAM, disk space and network bandwidth to a specific Virtual Machine [5, 6]. This approach has allowed the development of solutions such Software as a Service (SaaS) and Platform as a Service (PaaS), where services providers can quickly make available virtual machines with the required resources to their customers almost in no time, and not burden them with infrastructure management.

J. Mizera-Pietraszko et al. (Eds.): RTIS 2017, AISC 756, pp. 196–208, 2019.
https://doi.org/10.1007/978-3-319-91337-7_19

Sometimes, for servers' maintenance period, load balancing across nodes, fault tolerance and energy management on the datacenter, administrators are required to perform migration from one node to another. Accordingly, migration has become of the most important aspects of virtualization and even a powerful selling argument for many cloud services providers.

Operating Systems (OS) instances migration across different physical hosts is a useful feature for Cloud datacenters administrators. By performing migration while OS continue to operate, service providers can guarantee high performances with minimal application downtime. It can actually be so small so that a virtual machine user does not even notice it. However, as small as the downtime might be, it can cause some serious issues for time-sensitive applications such as online gaming, video streaming and critical web servers. Therefore, this degradation is simply not acceptable for these service and cause harm to the quality of service and service level agreement offered by the provider. Thus, it is vital to understand the migration procedure and techniques in order to integrate them efficiently and reduce their impact on the quality of service of the cloud applications [1].

This paper is organized as follow: Sect. 2 presents the direct migration mechanisms of VM and containers. Hybrid migration is illustrated in Sect. 3 where we describe the selected architecture to achieve complementarity between hypervisor-based VM and containers and how it positively affects the performances and management of the cloud. Section 4 presents the experimental part of our research to assess the performances of direct migration using hybrid virtualization. Conclusions and perspectives are described in Sect. 5.

2 Live Migration

2.1 Migration of Virtual Machine (VM)

Migration allows a clear isolation between Hardware and Software and optimize portability, fault tolerance, load balancing and low level system maintenance [7] (Fig. 1).

Fig. 1. Impact of live migration on user's request

VM migration can be classified into two categories [8]:

- Cold migration: when an administrator is planning to upgrade a piece of hardware or perform a maintenance task on a cluster, he/she must inform the applications owners (the customers) in advance about the maintenance schedule and its duration. Then, the hardware is turned off and back on only when the system and data are successfully restored. This approach is bother-some as it is not compliant with High Availability as the downtime is generally important.
- Live migration: it based on the process of transferring a virtual machine or a container while running from a physical node to another with a minimal interruption time. The migration of the physical host can be achieved by moving the state of processor, network and storage toward a new one and even the content of the RAM.

The process of transporting a virtual machine from a physical server A to a new one, B, consists of five steps [8, 9]:

Step 1: Reservation. Making sure that the destination server (B) has enough resources to host the VM to migrate and allocate them.

Step 2: Iterative pre-copy. Transferring data from memory pages of server (A) toward server (B). This action is repeated then only for the update pages (Dirtied) during the previous transfer.

Step 3: Stop-and-Copy. Halt the Virtual Machine on the source server (A) then copy the remaining data to destination server (B).

Step 4: Engaging. Send a notification message from the server (B) to (A) to inform about the end of migration. The VM instance is deleted from the source sever (A).

Step 5: Activation. Enabling the VM that has been successfully migrated to server (B).

2.2 Containers Migration

Containers technologies [10–13] is generally very interesting considering that it simplifies portability of cloud applications: it allows the execution of an application across multiple platform and platform without worrying about low level dependencies or OS compatibility. This explains their attractiveness for multiple businesses as they offer unprecedented flexibility for application deployment (Fig. 2).

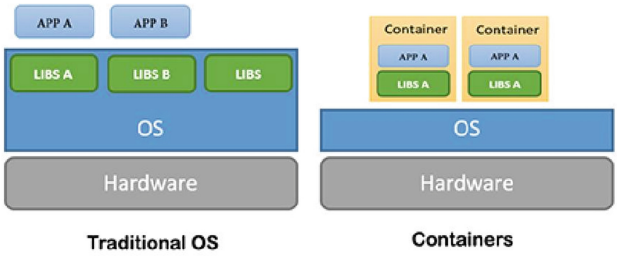


Fig. 2. Difference between deployment on traditional OS and containers

Unlike hypervisor-based virtualization, container-based virtualization is performed at a low-level system and does not aim to emulate a full hardware environment. It relies on the Linux Kernel capabilities to isolate applications. With this level of virtualization, multiple isolated Linux systems (containerized) can run on a control host while sharing a single instance of the OS kernel. Each container has its own processes and own network. The isolation is achieved thank to the name spaces. In fact, the processes of a container have unique identifier within their name space and cannot interact with those of another name space directly. A container can be seen as a set of processes (binaries, associated libraries, config files) that has its own lifecycle and dependencies and separated from other containers. Containerization is done today using tools like LXC (LinuX Containers) [11], Docker [12] or OpenVZ (Fig. 3).

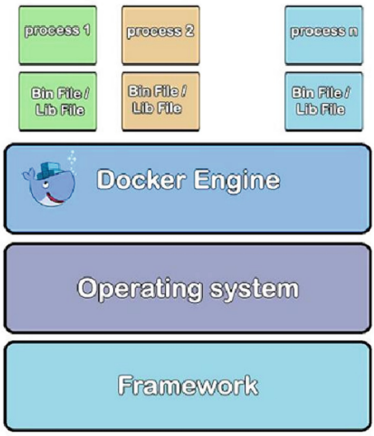


Fig. 3. Containerization is a set of processes sharing a namespace

Container migration is relatively a new technology on the market. Generally speaking, containerization is getting a lot of attractiveness from many companies thanks to its many benefits: almost no downtime maintenance, low effort for configuration and deployment, flexibility and the simplicity to achieve high availability. Unlike virtualization, containerization allows virtual instances (containers) to share a single host operating system, including binary files, libraries, or drivers. Thanks to that, it allows a server to potentially host many more containers than virtual machines. The difference in hosting can be considerable, so a server can accommodate 10 to 100 times more container instances than virtual machine instances. Because an application container does not depend on the OS, it is significantly smaller, easier to migrate or download, faster to backup or restore. Moreover, by isolating software packages from each other in containers, it [11–13] ensures a higher safety of sensitive applications.

Container migration is relatively complex, because containers are rather "processes" belonging to a namespace. Currently, container migration is based on the CRIU tool (Checkpoint/Restore In Userspace). CRIU performs this by freezing all process states

on the Host machine and then restores them in the destination machine according to the sequence diagram of Fig. 4.

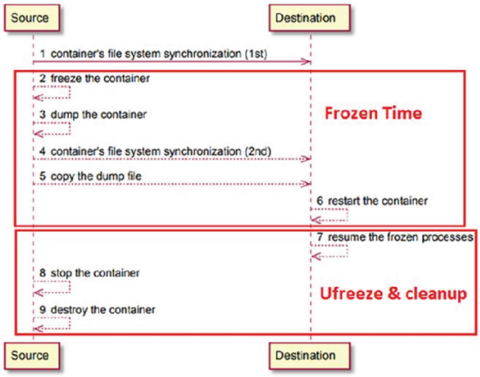


Fig. 4. The sequence diagram showing the live migration of the container

3 Hybrid Virtualization Using Container and VM

Because containers on the same physical host share the operating system kernel, a container security violation might compromise and affect other applications that share the same physical host. More precisely, because of the software nature of isolation in containers, unlike VM where it is of a hardware type, the security aspect still a bit of a challenge. Because of these aspects, service providers do not offer the same migration capabilities, as it is the case for virtual machines (Fig. 5).

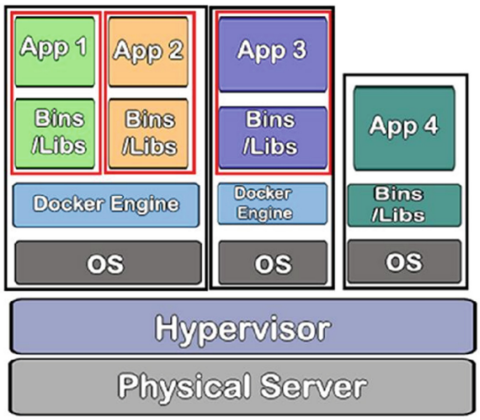


Fig. 5. Nesting of containers integrating applications in VMs

To remedy these problems, we will start by simply nesting a container in a virtual machine. This allows administrators to manage each container separately using a simple model: one container per VM. This allows using the existing virtualization management software and keeping the existing process. In addition, because each container relies on a VM as an additional layer of abstraction, administrators can avoid the security issues of multiple containers sharing the same OS kernel and maintain administrative consistency.

For Containerization, we focus mainly on Docker. This Hybrid architecture [14–19] raises questions about the efficiency of combining containers and VMs, and how to optimize the capacity usage of the hardware infrastructure. In other words, if virtual machines and Docker containers can coexist, and if so, can the Docker container services interact with VM services properly.

In fact, the experience described below shows that VM machines and containers can coexist and that container services can interact with the services of the VM virtual machine without any particular difficulty.

4 Experimental Setup

4.1 Hardware and Software Used

The experimental setup is as follow: for the hardware and software requirements, it consists of a client node (VLC media player) and two virtual servers (A and B) installed on VMware Workstation. We used a Laptop powered by a 2.4 GHz, i5 Intel processor, using a 8 GB of RAM. The operating system used is Ubuntu 14.04. We installed XenServer 7.0 [8] on servers A and B. Figure 6 shows the specifications used.

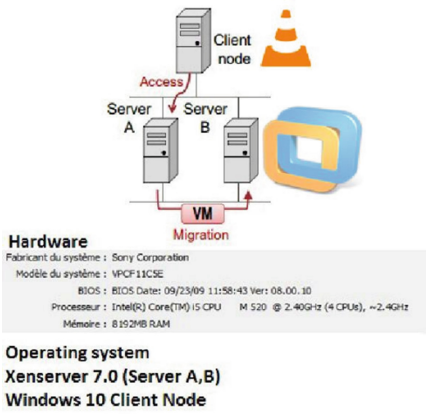


Fig. 6. Experimental configuration and hardware and software requirements

4.2 Experimentation

To study the impact of hybrid virtualization on live migration, we carried out the following scenarios:

Scenario 1: baseline. To have the ability to compare the impact of live migration of virtual machine VMs, we started with a machine that contains just the Ubuntu operating system in order to have a reference time. After the two servers were started, we started the VM in Server A and then migrated it from VM to Server B and recorded the processor and memory status during the operation.

Scenario 2: introducing VM virtualization. In this scenario, we have installed and configured Streaming Video Server [20, 21] in a VM machine. The Nginx server and the RTMP modules allow you to broadcast different types of video files. We started by installing the necessary software and dependencies and then configuring the server.

After we begin streaming the video, we start the video using VLC Player and we started the migration of the VM machine from server A to server B and recorded the state of the processor and memory during this operation. Figure 7a shows the specifications of Server A. In scenario 2 where we installed and configured a nginx [9] streaming server in a VM machine using Ubuntu 14.04 [22].

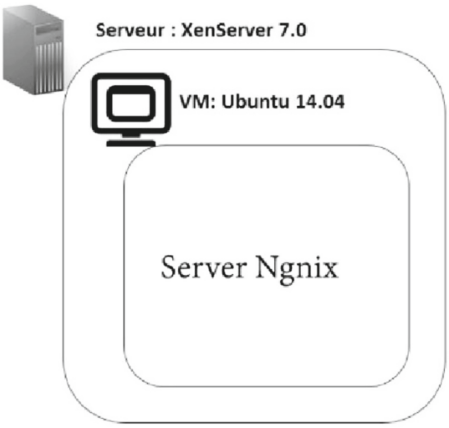


Fig. 7a. Specifications of Server A. In scenario 2

Scenario 3: Introducing Hybrid virtualization. In Scenario 3, we integrated the Docker container in which we installed the Nginx video streaming server in the VM machine.

After the running Docker Container, we start the video using VLC Player and we started the migration of the VM machine from server A to server B and recorded the state of the processor and memory during this operation. Figure 7b shows the specifications of Server A.

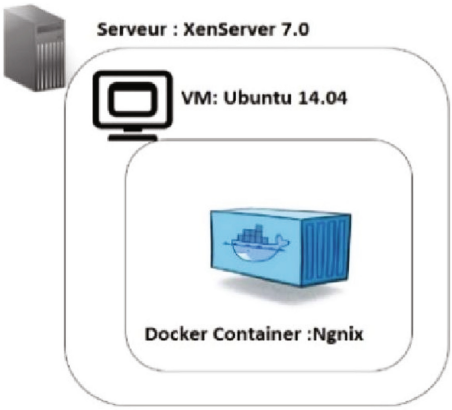


Fig. 7b. Specifications of Server A. In scenario 3

4.3 Results

Scenario 1. The migration of the VM machine between server A and B lasted 350 s. The virtual machine does not contain a streaming video server or a docker container. The following results show the status of the two servers A and B during the migration.

Server A status monitoring:

The CPU activity of the server A in all migration was normal except increments in the pre-copy iterative phase when the transfer of pages updated (dirtied) is repeated. The copy cycle stops when a number of Maximum iterations has been achieved.

The memory consumption remains maximum during the migration, and this decreases at the time of transfer from VM to the server B, where the memory used by the VM is released. Network traffic was low during all the migration. The migration time of scenario 1 is considered the reference migration time that will be used for comparison with the migration times of scenarios 2 and 3 (Fig. 8a and 8b).

Fig. 8a. Status of Server A

Server B status monitoring:

The CPU activity of Server B was average and almost stable throughout the migration and started to rise when the VM booted to Server B.

Memory activity was low during most migration time and started to rise when the VM was started on server B. Network traffic was low during all migration.

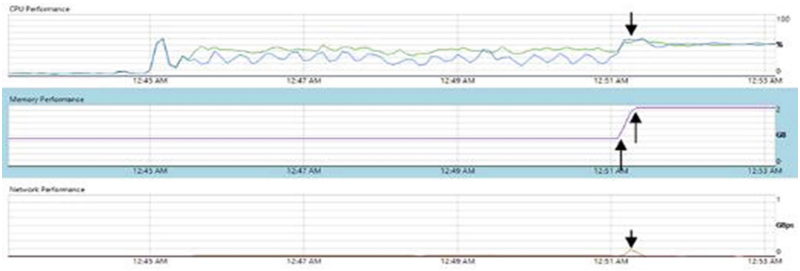


Fig. 8b. Status of Server B

Scenario 2. The migration between server A and B took 420 s. The Nginx video streaming server is installed and configured directly on the VM machine of server A. The following results shows the state of the two servers A and B during the migration (Fig. 9a and 9b).

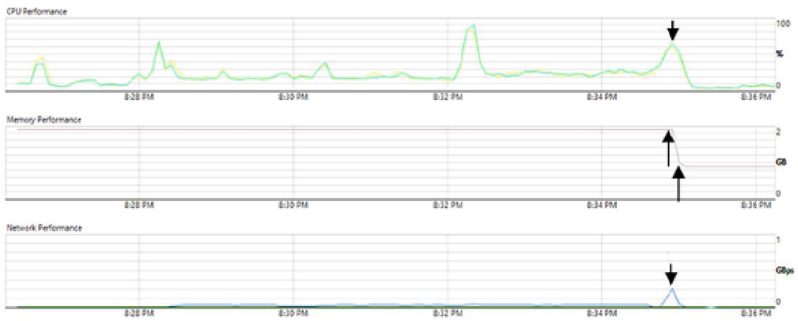


Fig. 9a. Status of Server A

Fig. 9b. Status of Server B

Server A. The CPU activity of the server A in all migration was normal except increment in the pre-copy iterative phase when the transfer of pages updated (dirtied) is repeated, The copy cycle stops when a number of Maximum iterations has been achieved.

The memory consumption remains maximum during the migration, and this decreases at the time of transfer from VM to the server B, where the memory used by the VM is released. Network traffic was low during all the migration.

Server B. The CPU activity of Server B was average and almost stable throughout the migration and started to rise when the VM booted to Server B. Memory activity was low during most migration time and started to rise when the VM was started on server B. Network traffic was low during all migration.

Scenario 3. Migration of the VM between server A and B lasted 390 s. The Nginx video streaming server is installed and configured in a Docker container nested in a VM machine. The following results show the status of the two servers A and B during the migration (Fig. 10a and 10b).

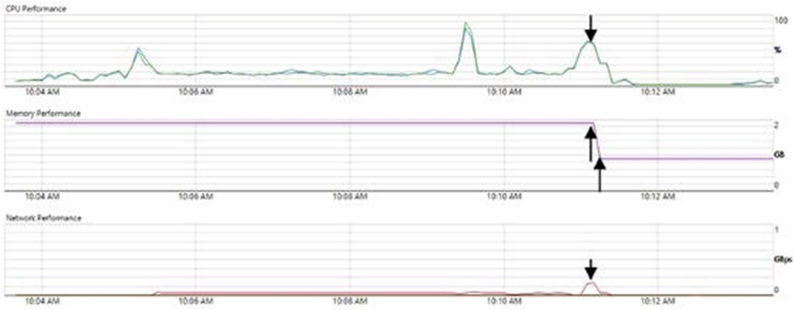


Fig. 10a. Status of Server A

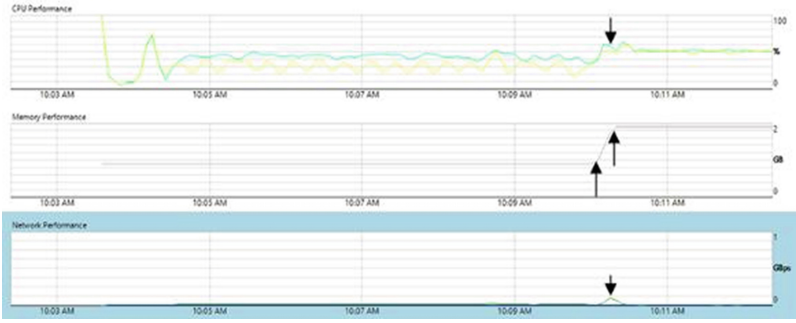
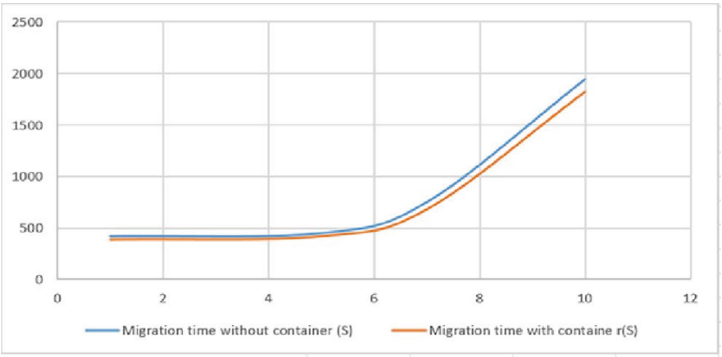


Fig. 10b. Status of Server B

In order to better compare the two scenarios we rewrite both scenarios 2 and 3 but increase the number of video streaming clients to study the impact of client overload on the server state and also the migration time (Fig. 11).



Number of users	1	5	7	10
Migration time without container (S)	420	450	750	1940
Migration time with containe r(S)	390	420	680	1820

Fig. 11. User overload impact on migration time

Comparing scenarios 2 and 3, we find that the migration time is lower in the case of scenario 3 than in scenario 2. This is mainly due to the nesting of the containers in the VM. We can explain this by the lack of a virtualization layer in the case of the containers, which translates into a rapid interaction with the operating system processes that affect the migration time. We also noticed a marked increase in network activity and use of CPU as mentioned in Fig. 12.

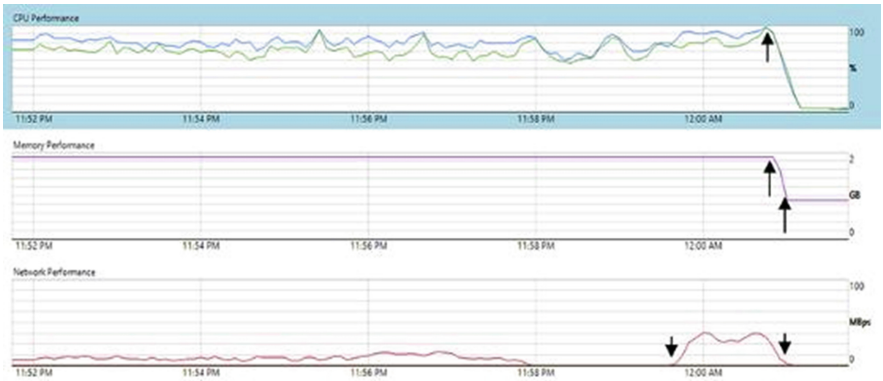


Fig. 12. Server A status with a 10 user load

5 Conclusion

In this work, we conducted a study on the value of Hybrid virtualization and its impact on direct migration. For this, we studied the direct migration of a monolithic application and precisely the video streaming installed in a container nested in a virtual machine, as such, application is sensitive to timing and performance. The results presented showed

the advantages of Hybrid virtualization especially in terms of total migration time. Indeed, the absence of a real virtualization layer in the case of containers results in a rapid interaction with the operating system processes, which affects positively the total migration time. As a perspective, we will study the contribution of the Hybrid architecture on the migration and the administration of layers of N-tiers applications (J2EE application) where each third party will be hosted in a container.

References

1. Vaquero, L.M., Rodero-Merino, L., Caceres, J., Lindner, M.: A break in the clouds: towards a cloud definition. Comput. Commun. Rev. **39**(1), 50–55 (2009). SAP Res., Belfast, United Kingdom, Telefonica Investig. y Desarrollo, Madrid, Spain
2. Site web de Xen Hypervisor. http://www.xen.org/
3. HyperV. https://msdn.microsoft.com/fr/library/hh846766(v=ws.11).aspx
4. VMware. http://www.vmware.com
5. Vaughan-Nichols, S.J.: New approach to virtualization is a lightweight. Computer **39**(11), 12–14 (2006)
6. Bachu, R.: A framework to migrate and replicate VMware virtual machines to amazon elastic compute cloud: performance comparison between on premise and the migrated virtual machine. Master thesis, Blekinge Institute of Technology (BTH) (2015)
7. Liu, H., He, B.: VMbuddies: coordinating live migration of multi-tier applications in cloud environments. IEEE Trans. Parallel Distrib. Syst. **26**(4), 1045–1205 (2015)
8. Kikuchi, S., Matsumoto, Y.: Impact of live migration on multi-tier application performance in clouds. In: 2012 IEEE Fifth International Conference on Cloud Computing. University of California at Berkley, USA (2009)
9. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live migration of virtual machines. In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation, vol. 2, pp. 273–286. USENIX Association (2005)
10. Kotikalapudi, S.V.N.: Comparing live migration between linux containers and kernel virtual machine: investigation study in terms of parameters. Master of Science in Computer Science, February 2017
11. Joy, A.M.: Performance comparison between linux containers and virtual machines. In: International Conference on Advances in Computer Engineering and Applications (ICACEA), pp. 342–346 (2015)
12. Li, W., Kanso, A.: Comparing containers versus virtual machines for achieving high availability. In: IEEE International Conference on Cloud Engineering (IC2E), 9–13 March 2015 (2015). https://doi.org/10.1109/ic2e.2015.79
13. Felter, W., Ferreira, A., Rajamony, R., Rubio, J.: An updated performance comparison of virtual machines and linux containers. In: IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp. 171–172 (2015). https://doi.org/10.1109/ispass.2015.7095802
14. Sahni, S., Varma, V.: A hybrid approach to live migration of virtual machines. In: IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), vol. 6 (2012)
15. Biancheri, C., Dagenais, M.R.: A hybrid approach to live migration of virtual machines. J. Cloud Comput. **5**, 19 (2016). ISSN 2192-113X
16. Bigelow, S.J.: Quelle différence entre conteneurisation et virtualisation? http://www.lemagit.fr/conseil/Quelle-est-la-difference-entre-la-conteneurisation-et-la-virtualisation

17. von Eicken, T.: Docker vs. VMs? Combining Both for Cloud Portability Nirvana, 02 September 2014. http://www.rightscale.com/blog/cloud-management-best-practices/docker-vs-vms-combining-both-cloud-portability-nirvana
18. Coleman, M.: Container and VMS Together, 8 April 2016. https://blog.docker.com/2016/04/containers-and-vms-together/
19. Hines, M.R., Deshpande, U., Gopalan, K.: Post-copy live migration of virtual machines. ACM SIGOPS Oper. Syst. Rev. **43**(3), 14–26 (2009)
20. Rattanaopas, K., Tandayya, P.: Performance analysis of a multi-tier video on demand service on virtualization platforms. In: International Computer Science and Engineering Conference (ICSEC), 23–26 November 2015 (2015). https://doi.org/10.1109/icsec.2015.7401437
21. Site Web du serveur vidéo Nginx. https://nginx.org/en/
22. Site web Système d'exploitation LinuxUbuntu 14.04. https://ubuntu-fr.org/