Performance analysis of Linux containers for high performance computing applications

Article *in* International Journal of Grid and Utility Computing · January 2017

DOI: 10.1504/IJGUC.2017.10009368

CITATIONS
5

READS
1,367

6 authors, including:

Stenio Fernandes
Federal University of Pernambuco
122 PUBLICATIONS **1,844** CITATIONS

SEE PROFILE

Jymmy Barreto
Federal University of Pernambuco
5 PUBLICATIONS **71** CITATIONS

SEE PROFILE

Patricia Takako Endo
University of Pernambuco
256 PUBLICATIONS **2,855** CITATIONS

SEE PROFILE

Performance analysis of Linux containers for high performance computing applications

The authors would like to thanks Inderscience Publishers and IJGUC to allow us to share this paper. Please refers to this paper as follows:

Beserra, D., Moreno, E.D., Endo, P.T., Barreto, J., Fernandes, S. and Sadok, D. (2017) 'Performance analysis of Linux containers for high performance computing applications', Int. J. Grid and Utility Computing, Vol. 8, No. 4, pp.321–329.

https://doi.org/10.1504/IJGUC.2017.088266

David Beserra* and Edward David Moreno

Universidade Federal de Sergipe,
Aracaju, SE, Brazil
Email: dw.beserra@gmail.com
Email: edwdavid@gmail.com
*Corresponding author

Patricia Takako Endo

Campus Caruaru,
University of Pernambuco,
Caruaru, PE, Brazil
Email: patricia.endo@upe.br

Jymmy Barreto, Stênio Fernandes and Djamel Sadok

Center of Informatics,
Federal University of Pernambuco,
Recife, PE, Brazil
Email: jymmyb@gmail.com
Email: sflf@cin.ufpe.br
Email: jamel@cin.ufpe.br

Abstract: Although cloud infrastructures can be used as High Performance Computing (HPC) platforms, many issues from virtualisation overhead have kept them almost unrelated. However, with advent of container-based virtualisation, this scenario acquires new perspectives because this technique promises to decrease the virtualisation overhead, achieving a near-native performance. In this work, we analyse the performance of a container-based virtualisation solution – Linux Container (LXC) – against a hypervisor-based virtualisation solution – KVM – under HPC activities. For our experiments, we consider CPU and (network and inter-process) communication performance. Results show that hypervisor type can impact distinctly in performance according to resource used by HPC application.

Kywords: cloud computing; HPC; LXC; container-based virtualisation; performance evaluation.

Reference to this paper should be made as follows: Beserra, D., Moreno, E.D., Endo, P.T., Barreto, J., Fernandes, S. and Sadok, D. (2017) 'Performance analysis of Linux containers for high performance computing applications', *Int. J. Grid and Utility Computing*, Vol. 8, No. 4, pp.321–329.

Biographical notes: David Beserra is a graduate student at the Federal University of Sergipe (UFS). His research interest is high performance computing, focusing performance evaluation of virtualised infrastructures.

Edward David Moreno received his PhD in Electrical Engineering at the University of São Paulo (USP). He is an Associate Professor at the Department of Computer Science, Federal University of Sergipe (UFS).

Patricia Takako Endo received her PhD of Computer Science from the Federal University of Pernambuco (UFPE) in 2014. She is a Professor at University of Pernambuco (UPE) since 2010. Her current research interests are cloud computing and resource management.

Jymmy Barreto is a graduate student in Computer Science at Federal University of Pernambuco (UFPE).

Stênio Fernandes received his PhD in Computer Science at the Federal University of Pernambuco (UFPE). He is an Associate Professor at the Center of Informatics (CIn), Federal University of Pernambuco.

Djamel Sadok received his PhD of Computer Science at the University of Kent at Canterbury, UK, in 1990. He is a member of staff at the Computer Science Center of the Federal University of Pernambuco (UFPE) since 1993. His research interests include communication systems, access networks, security, cloud computing and traffic classification.

This paper is a revised and expanded version of a paper entitled 'Performance analysis of LXC for HPC environments' presented at the '2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)', Blumenau-SC, Brazil, 8–10 July 2015.

1 Introduction

High Performance Computing (HPC) is a generic term for applications that are computationally intensive or data intensive in nature (Simons and Buell, 2010; Beserra et al., 2014): ranging from floating-point computation in micro-controllers to 3D real-time medical images and simulations of large-scale structures, taking like a good example the Universe (Vogelsberger et al., (2014). Commonly, HPC applications are executed in classical dedicated *cluster infrastructures* in which resources are locally owned with private access and are composed of structures with static configurations (Beserra et al., 2015b). Currently, we can also host HPC applications on Cloud infrastructures, such as Nimbus (Keahey et al., 2007) and Neblina (Fernandes et al., 2012) platforms.

HPC applications have used *Cloud Computing* infrastructure since the advent of scientific Clouds (Keahey et al., 2007) and virtualised computer clusters (Napper and Bientinesi, 2009) owing to its facility to rent, to manage and to allocate resources according to demand. As benefits arising from Cloud Computing to HPC, we can highlight high availability (HA), operating system (OS) customisation, elasticity, and cost reduction of resource maintenance (Ye et al., 2010; Napper and Bientinesi, 2009).

In Cloud Computing environment, *virtualisation* is a key technology for enabling its operation and, despite many advantages, virtualisation commonly suffers performance penalties due to resource sharing, always presenting worst performance than non-virtualised scenarios (Regola and Ducom, 2010). Trying to overcome this performance issue, *container-based virtualisation* solutions (such as Linux-VServer, OpenVZ and Linux Containers [LXC]) have been proposed and analysed, and, according to Xavier et al. (2013), '*this type of virtualization offers a lightweight virtualization layer, which promises a near-native performance*'.

Some works in the literature have done performance analysis comparing virtualisation technologies for HPC, such as Ye et al. (2010), Regola and Ducom (2010), Younge et al. (2011), and Xavier et al. (2013); however, our main contribution is to *analyse the suitability of LXC for HPC applications*, comparing it with a native (non-virtualised) environment, and a traditional hypervisor-based solution with support for hardware-assisted virtualisation and drivers for network device paravirtualisation, Kernel-based Virtual Machine (KVM).

For this, we will conduct experiments with traditional benchmarks considering different VM allocation possibilities: many VMs on the same host competing or coopering for resource usage, and VMs allocated in distinct hosts without resource sharing.

This work is structured as follows: Section 2 presents virtualisation basic concepts needed to understand this work; Section 3 describes the related works; Section 4 presents our main goals and the methodology used to perform our experiments; we present and discuss the results in Section 5; and finally, we describe our considerations and delineate future works in Section 6.

2 Virtualisation background

Currently, we have four approaches to provide resource virtualisation: full virtualisation, hardware-assisted virtualisation, paravirtualisation, and operating-system-level (OS-level) virtualisation. The *full virtualisation* allows a guest OS without kernel modifications. However, it imposes high overhead in VM performance. This overhead can be mitigated using *hardware-assisted virtualisation*, a set of specific instructions to virtualisation that is present in almost all processors and in some I/O elements (Nussbaum et al., 2009).

When using *paravirtualisation* approach, hardware devices are accessed through special paravirtualised drivers, obtaining a better performance when compared to full virtualisation, even when it is assisted by hardware (Ye et al., 2010). However, the guest OS kernel must be modified in order to provide new system calls. This modification increases the performance because it reduces the CPU consumption; however, at the same time, it reduces the security and increases the management difficulty.

The *OS-level virtualisation* creates containers that allow processes have their own isolated resources without hardware emulations, accessing hardware resources in a direct way. Each container runs its own OS and file system, but shares the kernel with other containers and the host OS (Xavier et al., 2013). According to Li et al. (2015), container can be considered '*as slicing the operating system into smaller units that can host their own applications with their own networking stack and computing resources*'.

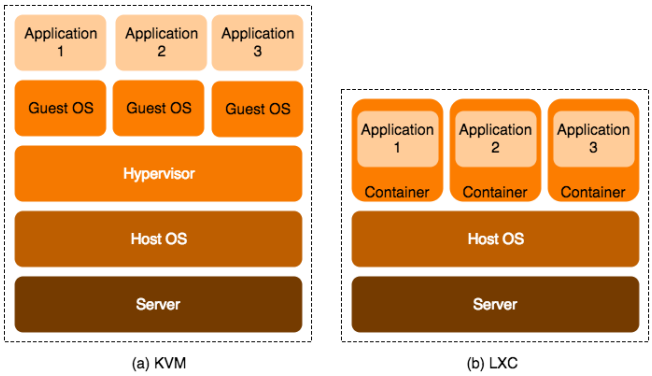
In this work, our main objective is to analyse OS-level virtualisation performance against other virtualisation technique. The virtualisation solutions used in this work are presented in details in the next subsection and the justifications for our choice are presented in Section 3.

2.1 KVM and LXC

KVM is an open source virtualisation tool integrated to Linux kernel. This virtualisation approach takes advantage of the Linux kernel development as own evolution

(Nussbaum et al., 2009). In KVM, the I/O and network management is done by a QEMU modified version. I/O requests done by a VM are forwarded to the QEMU that redirects them to the host OS (see Figure 1 (a)).

Figure 1 KVM and LXC (see online version for colour)



Linux Container (LXC) is a lightweight virtualisation mechanism that does not require emulation of physical hardware. The main usage of LXC is to run a complete copy of the Linux OS in a container without the overhead of running a level-2 hypervisor (Figure 1 (b)). The container shares the kernel with the host OS, so the processes and file system are completely visible from the host. On the other hand, the container only sees its file system and process space. The LXC takes the CGroups (control group) resource management facilities as its basis and adds POSIX file capabilities to implement processes and network isolation (Oracle Corporation, 2014).

3 Related works

Some works have already evaluated virtualisation tools for HPC, such as Nussbaum et al. (2009), that compared KVM and Xen taking into consideration metrics, such as CPU and network performance, as well as the CPU consumption by VMs during tests. Regarding network performance, the authors considered a common scenario in Cloud Computing environments, in which VMs are sharing the same host and sending a great number of packets. In this scenario, KVM presented aggregated bandwidth better than Xen because KVM network emulation has a better host CPU utilisation when processing data. Authors also observed that KVM and Xen have similar performance when considering different network operations (sending and receiving) and their performances are similar to the native environment.

Younge et al. (2011) analysed the performance of open-source hypervisors – Xen, KVM, and VirtualBox – running some benchmarks in virtual clusters. They created a hypervisor ranking for HPC, concluding that KVM and VitualBox present the best global performance and management facility, with KVM excelling in regard to computation capability and memory expansibility. Authors also observed that, different from Xen, KVM presents little performance oscillations, that is considered a key characteristic in Cloud environments. But, the authors did not specify if Xen was using its paravirtualisation instructions or not. Beserra et al. (2015b) expanded work done by Younge et al. (2011) and performed performance comparison between KVM and VirtualBox in cluster environments and noticed that KVM performs better than VirtualBox in all tests, and is more suitable for large-scale HPC hypervisor-based virtualised infrastructures.

However, these works did not consider OS-level virtualisation technologies based on containers. Containers can offer advantages for HPC, such as a short bootstrap time, and a less disk space usage. Seo et al. (2014) noted containers present a bootstrap duration 11 times smaller than KVM, and the disk usage is also smaller; while 50 VMs (KVM running Ubuntu) can be stored in a disk with 500Gb, more than 100 containers use less than a half of this space. Beyond that, container technology can also improve the service availability; for instance in Guedes et al. (2014) was achieved a five 9's availability in a web cache cluster scenario with 1:1 VM and disk over container-based OS virtualisation.

On the other hand, Che et al. (2010) compared the performance of OpenVZ, Xen, and KVM using VMs as web servers. OpenVZ presented better results than native in some situations; however, the KVM paravirtualisation instructions were not activated. This configuration could collaborate to KVM inferior performance, since KVM with paravirtualisation instructions can obtain better performances than Xen (Nussbaum et al., 2009; Younge et al., 2011).

Xavier et al. (2013) innovated when analysing the performance of main OS-level virtualisation solutions running HPC applications. They compare OpenVZ, LXC, and VServer against Xen performance considering CPU, disk, and network. All containers solutions and Xen presented an equivalent processing capacity, with all solutions reaching performance close to a native environment. The network performance analysis considered bandwidth and delay according to packet size and results indicated that LXC and VServer have best performance, with smaller delay and bigger bandwidth for any size of packet. *Since HPC applications were tested, thus far, LXC demonstrates to be the most suitable of the container-based systems for HPC* (Xavier et al., 2013).

Felter et al. (2014) compared the performance of KVM and a Linux container-based solution, Docker, regarding to non-virtualised environments. They evaluated CPU, memory, storage and networking resources, and observed Docker performance is equal to or better than KVM in all cases they tested. Both technologies introduce negligible overhead for CPU and memory performance; however, they need some improvements to support I/O intensive applications. Their results corroborate those of Xavier et al. (2013).

For our work, we decided to use *KVM* because it has the best performance and is the best hypervisor-based virtualisaton solution according to literature (Younge et al., 2011; Felter et al., 2014), and *LXC* because it can become the de facto standard to system containerisation, with the

possibility to converge with OpenVZ to compose the same tool (Dua et al., 2014).

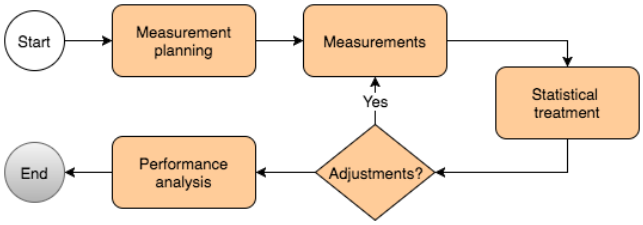
We extend the work done by Nussbaum et al. (2009), verifying how resource sharing among multiple VMs can impact on the performance, as well as analysing the CPU usage by guests (containers and VMs) according to different resource types.

We also analyse the network performance in function of size packets in a similar way to Xavier et al. (2013) and, for all experiment results, we take in consideration statistics analysis (such as variation and standard deviation) – aspect that is not contemplated in Xavier et al. (2013). Additionally, different from all previous works, we also evaluate the intra-node inter-process communication performance. In this way, we fulfil some lacks of existing works in literature and contribute to the state-of-the-art evolution.

4 Experimental design

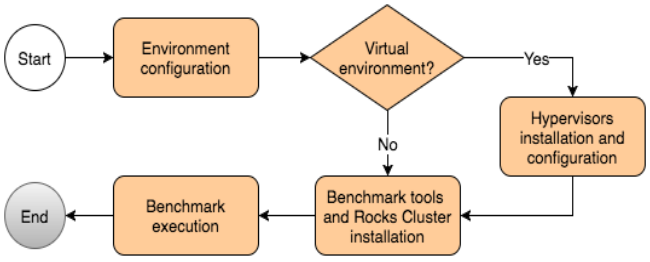
In this section, we describe the experimental design used to evaluate both virtualisation technologies. We adopted the methodology used in Sousa et al. (2012) to execute our measurements based on four main activities (Figure 2): first, it is important to planing all measurement experiments that will be done with KVM and LXC, defining how our measurement should be performed in order to collect metrics we are interested. In this activity, we also need to decide which benchmark tool we will use and how we will store the measured data.

Figure 2 Measurement methodology, adapted from (Sousa et al., 2012) (see online version for colour)



The measurement activity is explained in Figure 3. And the statistical treatment activity applies statistical methods to provide accurate information about our experiments.

Figure 3 Measurement activities, adapted from (Sousa et al., 2012) (see online version for colour)



4.1 Infrastructure

The experiments were executed in two HP EliteDesk 800 G1 (F4K58LT) computers equipped with Intel Core i7-4770 processors operating in 3.4 GHz, with 8 GB DDR3 SDRAM memory operating in 1600 MHz. The processor used has a set of specific instructions for virtualisation: VT-x technology. We used Gigabit Ethernet network adapters and switches to interconect the servers when needed. Both hosts and clusters used in this experiment were configured with Ubuntu 14.04.2 LTS 64 bit.

4.2 Metrics and benchmarking tools

We chose to analyse the performance of classical HPC performance metrics: CPU and communicating capacity (intra-node and network-based communications). For each metric, we apply a specific benchmark, described in next subsections.

4.2.1 CPU performance and HPL tool

To analyse the processor performance, we use the *High Performance Linpack (HPL)* benchmark. HPL measures float point operations per second (Flops) done by a computational system during a linear equations system resolution (Younge et al., 2011). We chose HPL tool because it is the default benchmark used by the TOP500 supercomputing site to elaborate a semiannual list containing the most powerful supercomputers of the world (Napper and Bientinesi, 2009).

4.2.2 Communicating performance and NetPIPE tool

The main goal when analysing the inter-proccess communication performance is to understand how the communication bandwidth and delay are impacted by the additional virtualisation layer. For this, we choose the *Network Protocol Independent Performance Evaluator (NetPIPE)* tool that monitors network overheads using protocols, like TCP, UDP and MPI. It performs simple ping-pong tests, sending and receiving messages of increasing size between a couple of processes, whether across a cluster connected by a Ethernet network or within a single multicore system.

4.3 Evaluated scenarios

In our experiments, the LXC performance was compared against KVM and native environments, considering the following goals:

1 To determine the overhead caused by virtualisation on CPU-bound applications performance;

2 To determine the overhead caused by resource sharing on performance of the shared resources in function of the sharing type; and

3 To determine how virtualisation affects the inter-process communication performance, considering:

   (a) Communication using only intra-node communication mechanisms;

   (b) Communication using a physical network interface.

All benchmarks were executed 32 times, for all tests, environments and conditions. In all tests, the two measurements with higher and lower values were excluded as outliers; with the other 30, we computed the media and standard deviation. Next, we describe each environment detail according to the goals aforementioned.

All environments were implemented in KVM and LXC and, even the number of nodes varies, all they had the same number of CPU cores and RAM. All environments used entire available resources and were implemented in a single server. The Native environment was composed of only one physical node with four processor cores and 3096 MB of memory; the Virtual-1 had 1 VM with four cores and 3096 MB of memory; the Virtual-2 had two VMs with two cores per VM and 1536 MB of memory per VM; and the Virtual-4 had four VMs with one core per VM and 768 MB of memory per VM.

Table 1 Single-server environments

Environment	*Nodes*	*Cores/node*	*Memory/node (MB)*
Native	1	4	3096
Virtual-1	1	4	3096
Virtual-2	2	2	1536
Virtual-4	4	1	768

To reach all of the goals targeted, we evaluated the environments described in Table 1, taking into consideration two possible scenarios of execution:

1 There is no communication between benchmark processes in execution;

2 The processes communicate with each other in a cooperative way.

4.3.1 Virtualisation overheads on CPU-bound applications performance

To evaluate the CPU performance when there is no interprocess communication, we instantiated one copy of HPL benchmark for each available processor. For instance, in the Native environment, we instantiated four HPL copies, each one executing in a distinct processor; while in the Virtual-4 environment, each VM had one HPL copy. So, as we had four VMs, we also had four HPL copies running. Here, we observed the aggregated performance (the sum of all individual HPL copies). On the other hand, when they were working in a cooperative way, using explicit interprocess communications, we ran only one HPL instance, using all available processors. The VMs in Virtual-2 and Virtual-4 environments were grouped in a cluster, with one HPL copy in execution.

4.3.2 Effects of resource sharing on performance

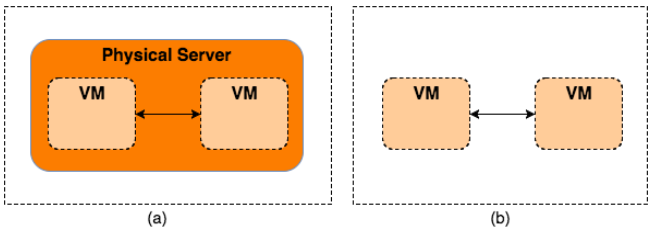
In this experiment we ran a copy of HPL benchmark in a VM of each type used in Virtual-2 and Virtual-4 environments. The results obtained in these tests were compared with those obtained individually by each VM of the same type when running concurrently with other VMs the same type. In theory, equal VMs have the same performances in the same activities. This test is an expansion of Beserra et al. (2015a) work.

It is important to ensure that two users who contracted a particular Cloud service are able to dispose the same performance. If the service (in this case instances of virtual resources hosted on the same server) is not provided with a constant performance for all customers, then it does not provide a good quality of service (QoS), which implies negative impacts for users and in judicial problems for Cloud providers. So, it is imperative that the OS host be able to allocate resources equally among the VMs (Nussbaum et al., 2009).

4.3.3 Virtualisation overheads on communication performance

We used NetPIPE to verify the occurrence of virtualisation overheads in intra-node inter-processes communications. First, we ran NetPIPE in Native and Virtual-1 environments; then, we ran NetPIPE in pairs of processes running in Virtual-2 environment, with each process in a distinct VM, to verify the virtualisation effects in inter-processes communication between VMs hosted in the same server. In this experiment, VMs did not use physical network, they used internal server mechanisms, as shown in Figure 4 (a). This experiment is relevant since the internal communication mechanisms can vary according to virtualisation approach employed, and it can also influence the performance.

Figure 4 Scenario (a) VMs (or containers) communication inside the same physical host and (b) physical network (see online version for colour)



The physical network performance is a critical component in cluster environments. When a communication enfolds different servers or VMs hosted in different servers, the physical network is used, increasing the communication delay and overhead. To evaluate the network performance, we executed NetPIPE with processes allocated in different physical servers (Figure 4 (b)), measuring the performance according to packet size (Turner et al., 2003). To determine how the physical network virtualisation affects real applications, we also ran the HPL benchmark in virtual clusters hosted in distinct servers.

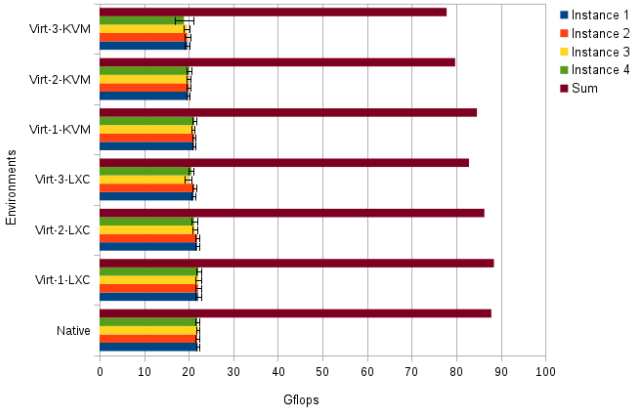
5 Results

This section presents the results grouped according to our goals that we obtained from benchmark tests described in the previous section.

5.1 Virtualisation overheads on CPU-bound applications performance

In this subsection, we present the results of HPL benchmark executed in a single server. Figure 5 shows results of CPU performance when executing HPL benchmark with no interprocess communication. We can observe LXC presented aggregated performance higher than KVM in Virtual-1 environment, and close to Native environment. Statistically, all variations were irrelevant (lower than 5%) for all environments and virtualisation technologies.

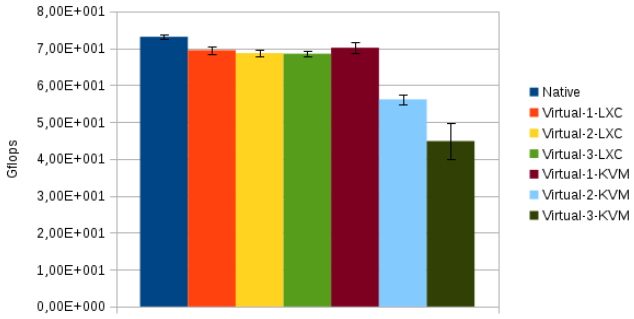
Figure 5 HPL aggregated performance (same host)



When we divided physical resources in multiple virtual environments, both virtualisation technologies presented a worse performance. This gradual decreasing affects KVM more poignantly because is needed to maintain multiple and isolated systems in simultaneous execution, increasing the overhead in host CPU. During Virtual-4 execution, for instance, the overhead of host CPU exceeds 100%, while LXC maintains the CPU usage within availability limits. These results indicate that, despite VT-X instructions set, administrative controls to provide emulation of CPU instructions and resource isolation between systems cause considerable overhead on performance.

Figure 6 shows the CPU performance results when MPI processes are working in a cooperative way. We can note LXC environments were constant and close to native, even when resources were shared among multiple containers. It was possible because the host system needs few resources to create and maintain a container (Xavier et al., 2013; Felter et al., 2014). On the other hand, KVM environments presented gradual performance reduction while we increased the number of VMs sharing the same host. It occurs because KVM needs more resources to maintain VMs, and when the system is divided in more than one logic node, processes will communicate by using network protocols, increasing the processing and the communication latency.

Figure 6 HPL local clusters performance (same host)



5.2 Effects of resource sharing on performance

In this subsection, we present the results of HPL benchmark executed in the environments running concurrently some instances of virtual abstractions. Figure 7 shows the CPU performance benchmark results of one virtual resource instance running isolated against two instances of the same type running simultaneously at the same host server. Figure 8 shows a similar comparison, but with four instances of the same type.

Figure 7 Effects of resource sharing on performance with two instances running concurrently

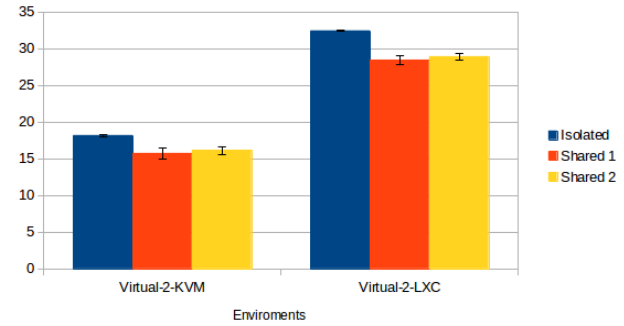
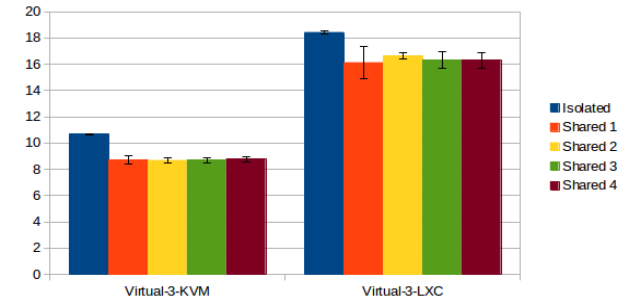


Figure 8 Effects of resource sharing on performance with four instances running concurrently



In both cases, we can observe the occurrence of some performance contraction in virtualised shared resources, with different reduction rates for each virtualisation tool. In the case of KVM, when running two VMs of the same type simultaneously, the performance of each VM decreased about 11.5% in relation to same VM running alone. When running simultaneously four VMs of the same type that we used in the Virtual-4 environment, the reduction rate relative to a isolated VM jumped to 20%.

This probably occurred due to an overhead in processor of the host server, which provided and maintained multiple virtual resources at the same time. In this case, to avoid performance losses and a subsequent service quality below of predetermined levels in the Service Level Agreement (SLA) is necessary to provide a threshold for native host resource usage flexible in function of the amount of allocated VMs; it is possible to develop an appropriate SLA that will allow both providers and users enjoy a stable service with respect to performance.

On other hand, when we repeated the experiments for LXC, we observed a fixed value of 11.5% pair to both sharing scenarios. This is good, because it allows that any problems regarding performance can be solved with a static threshold, enabling the creation of a more simply SLA.

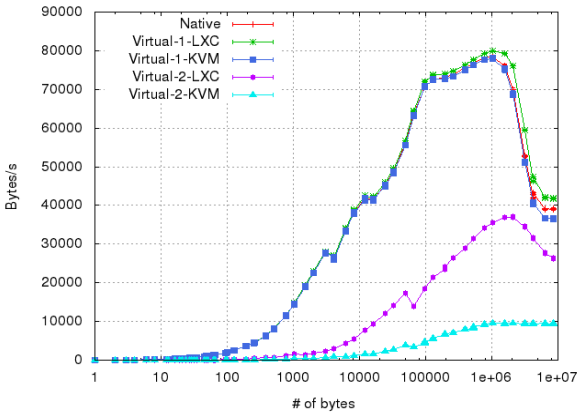
Regarding to the performance variation on the virtualised resources, KVM behaved in a similar way to that observed in previous tests and showed a stable performance with no significant oscillations. The LXC, on the other hand, showed higher variations, but still presenting only a few statistical significance.

This indicates that both virtualisation technologies are useful for balancing virtual resources if the environment manager adds an adequately threshold policy regarding to physical server resources usage. Probably, if more resources were allocated to the virtual server at a point where the physical resources are overloaded, it will be very unlikely that the OS host has the capability to split the resources equally among multiple virtual environments, as observed in previous experiments (Mello et al., 2010).

5.3 Network performance

Figure 9 shows results of NetPIPE execution in a single server. We can observe that in Virtual-1 environment both virtualisation technologies achieved communication bandwidth similar to the one achieved by Native environment, and the performance variations were insignificant. However, when we divided the physical server in multiple virtual environments, the communication bandwidth achieved was reduced for both virtualisation tools, especially KVM.

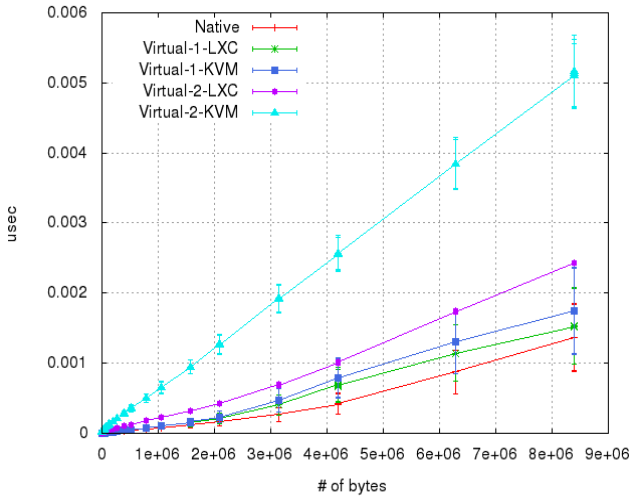
Figure 9 Inter-process communication bandwidth (same host)



Since Virtual-2 environment was configured in a cluster, it was needed to use network protocols to perform the communication between NetPIPE processes, increasing the CPU usage in the host, and also contributing to decreased bandwidth. For KVM, we can note an additional reduction of bandwidth; it occurred because the host uses more CPU to emulate two network interfaces simultaneously, as well as whole virtual communication environment in an isolated way.

This tendency is endorsed when we observe delay communication results in Figure 10. We found high values of delay when we have more requests to process. With exception of Native, all environments presented considerable variations for both virtualisation tools.

Figure 10 Inter-process communication latency (same host)



Figures 11 and 12 show results of NetPIPE execution in two servers. All environments presented similar performance with little variations for communication bandwidth, as well as for delay.

Figure 11 Inter-process communication bandwidth (cluster)

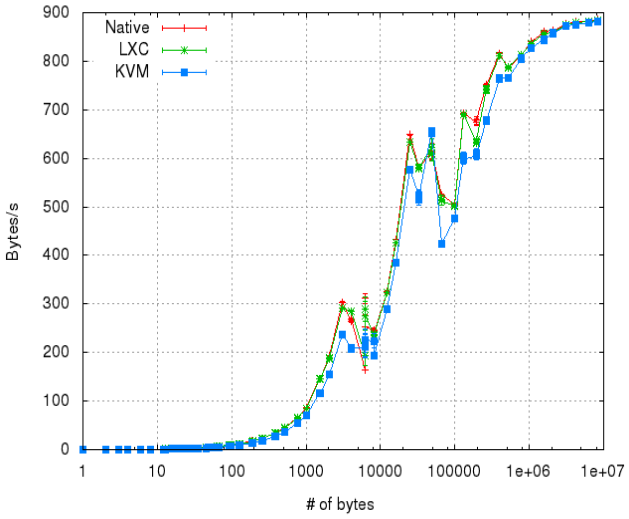
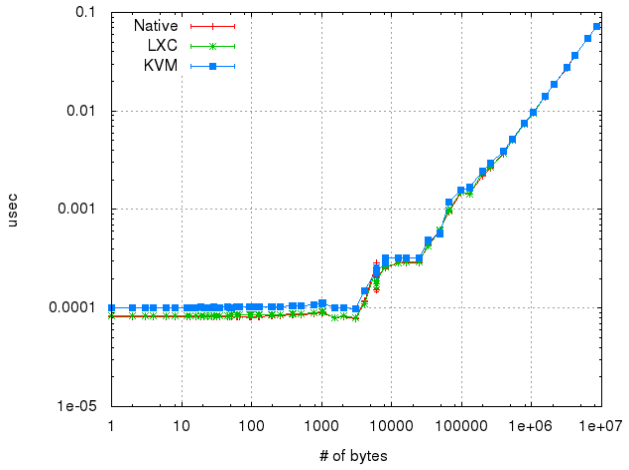
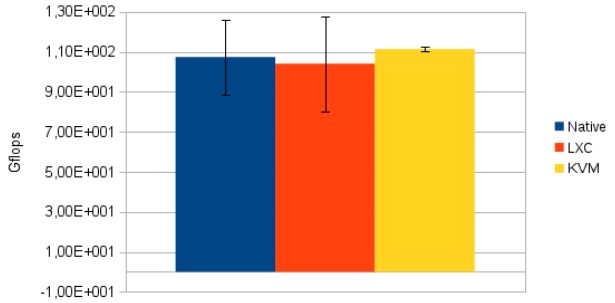


Figure 12 Inter-process communication latency (cluster)



To finish our communication analysis, we verified how network virtualisation affects the HPC application performance. For that, we executed HPL in cluster environment with two servers, as shown in Figure 13. Both environments presented similar performance; however, KVM cluster showed less variations than LXC.

Figure 13 HPL cluster performance (using network)



5.4 Discussion

Container-based solutions are a good alternative to overcome the overhead issues from virtualisation. They can be seen as more flexible tools for packaging, delivering and orchestrating both software services and applications (Peinl et al., 2016).

However, during our experiment executions, we have faced some open issues, specifically for LXC. We observed LXC may not provide sufficient *isolation* at this time, allowing guest systems to compromise the host system under certain conditions. Moreover, LXC-halt times-out, and when we tried to upgrade to 'Jessie' it broke the container.

This matches work done by Li et al. (2015), who say that containers, as opposed to VMs, have many advantages in terms of performance because of their *lightweight-ness*, but there still are many gaps in containerised platforms that need to be filled.

However, although containers are still facing some technical issues, '*we recently see major players in virtualization and Linux distributions realizing the potential and perhaps the threat of containers and hence partnering with, and supporting de-facto containerization solutions*' (Li et al., 2015). In the same way some Cloud Computing middlewares, such as OpenStack, are proving support for LXC usage in Cloud environments (Sefraoui et al., 2012), inclusive in Cloud federation contexts (Khan et al., 2014).

In this way, we strongly believe that containers can improve the Cloud performance and facilitate resource management, but at same time, it is necessary to work to mitigate open and crucial issues, such as resource isolation and security.

6 Conclusions and future works

This paper presented a performance evaluation between two virtualisation tools: KVM and LXC. According to our experiments, we observed *LXC is more suitable for HPC than KVM*. Considering a simple case of use, when virtual environments use all server resources for only one VM or container, both systems presented similar processing performance. However, in a more complex (and more common) scenario, in which physical resources are divided among multiple and logic environments, both decrease their performances, especially KVM.

For cluster environments, in which processes work in a cooperative way and there is communication between processes, we can see the difference more clearly. Regarding variations in applications performance, LXC was better again, presenting fewer performance fluctuations for most of tests.

This is an important aspect to be considered, because in Cloud Computing environments, if customers are paying for a specific Cloud service, then one expects that all customers receive services with equal performance (or at least with a minimal level previously agreed), otherwise Cloud provider can suffer agreement penalty and, consequently, financial losses (Napper and Bientinesi, 2009).

As future works, we plan to evaluate these virtualisation technologies regarding the I/O performance for traditional Hard Drives (HD), as well as for SSD devices, and to Graphical Processing Unit (GPU) for single server and cluster environments. Other aspect that can be developed in future is to analyse how federated and/or hybrid Clouds could be benefited by sharing their resources to execute HPC applications.

References

Beserra, D., Karman, R., Camboim, K., Araujo, J., Borba, A. and de Araujo, A.E.P. (2014) 'Análise do Desempenho de Sistemas Operacionais Hospedeiros de Clusters Virtualizados com o VirtualBox', *Proceedings of XXXII Brazilian Symposium on Computer Networks – XII Workshop in Clouds and Applications*, SBC, Brazil, pp.3–15.

Beserra, D., Moreno, E., Endo, P., Barreto, J., Sadok, D. and Fernandes, S. (2015a) 'Performance analysis of LXC for HPC environments', *2015 Ninth International Conference on in Complex, Intelligent, and Software Intensive Systems (CISIS)*, IEEE, Blumenau, Brazil, pp.358–363, doi:10.1109/CISIS.2015.53.

Beserra, D., Oliveira, F., Araujo, J., Fernandes, F., Araújo, A., Endo, P., Maciel, P. and Moreno, E. (2015b) 'Performance analysis of hypervisors for HPC applications', *2015 IEEE International Conference on in Systems, Man and Cybernetics (SMC)*, IEEE, Kowloon, China, pp.846–851.

Che, J., Yu, Y., Shi, C. and Lin, W. (2010) 'A synthetical performance evaluation of openVZ, Xen and KVM', *2010 IEEE Asia-Pacific Services Computing Conference (APSCC)*, IEEE, Hangzhou, China, pp.587–594.

Dua, R., Raja, A.R. and Kakadia, D. (2014) 'Virtualization vs containerization to support PaaS', *2014 IEEE International Conference on Cloud Engineering (IC2E)*, IEEE, Boston, MA, USA, pp.610–614.

Felter, W., Ferreira, A., Rajamony, R. and Rubio, J. (2014) *An Updated Performance Comparison of Virtual Machines and Linux Containers*, IBM Technical Report RC25482 (AUS1407-001), Austin Research Laboratory, Austin, TX.

Fernandes, J., Barbosa, J., Shculze, B. and Mury, A. (2012) 'Integration of cloud services in support to tests, simulations and knowledge dissemination with cluster environments', *Proocedings of 13th Symposium on Computer Systems (WSCAD-SSC 2012)*, IEEE, Petropolis, Brazil, pp.72–79.

Guedes, E., Silva, E. and Maciel, P. (2014) 'Performability analysis of I/O bound application on container-based server virtualization cluster', *2014 IEEE Symposium on Computers and Communication (ISCC)*, IEEE, Funchal, Madeira, Portugal, pp.1–7.

Keahey, K., Freeman, T., Lauret, J. and Olson, D. (2007) 'Virtual workspaces for scientific applications', *Journal of Physics: Conference Series*, Vol. 78, No. 1, p.012038.

Khan, A.M., Selimi, M. and Freitag, F. (2014) 'Towards distributed architecture for collaborative cloud services in community networks', *2014 International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, IEEE, Salerno, Italy, pp.1–9.

Li, W., Kanso, A. and Gherbi, A. (2015) 'Leveraging Linux containers to achieve high availability for cloud services', *2015 IEEE International Conference on Cloud Engineering (IC2E)*, IEEE, Tempe, AZ, USA, pp.76–83.

Mello, T., Schulze, B., Pinto, R. and Mury, A. (2010) 'Uma análise de recursos virtualizados em ambientes de HPC', *Proceedings of XXVIII of Brazilian Symposium on Computer Networks – IX Workshop in Clouds and Applications*, SBC, Brazil, pp.17–30.

Napper, J. and Bientinesi, P. (2009) 'Can cloud computing reach the top500?', *Proceedings of the Combined Workshops on Unconventional High Performance Computing Workshop Plus Memory Access Workshop (UCHPC-MAW)*, ACM, Ischia, Italy, pp.17–20.

Nussbaum, L., Anhalt, F., Mornard, O. and Gelas, J-P. (2009) 'Linux-based virtualization for HPC clusters', *Proceedings of the Montreal Linux Symposium 2009*, Montreal, Canada.

Oracle Corporation (2014) *Oracle VM VirtualBox: User Manual*. Available online at: https://www.virtualbox.org/manual/UserManual.html (accessed on 26 August 2014).

Peinl, R., Holzschuher, F. and Pfitzer, F. (2016) 'Docker cluster management for the cloud-survey results and own solution', *Journal of Grid Computing*, Vol. 14, No. 2, pp.265–282.

Regola, N. and Ducom, J-C. (2010) 'Recommendations for virtualization technologies in high performance computing', *IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)*, IEEE, Indianapolis, IN, USA, pp.409–416.

Sefraoui, O., Aissaoui, M. and Eleuldj, M. (2012) 'OpenStack: toward an open-source solution for cloud computing', *International Journal of Computer Applications*, Vol. 55, No. 3, pp.38–42.

Seo, K.T., Hwang, H.S., Moon, I.Y., Kwon, O.Y. and Kim, B.J. (2014) 'Performance comparison analysis of Linux container and virtual machine for building cloud', *Advanced Science and Technology Letters Journal*, Vol. 66, pp.105–107.

Simons, J.E. and Buell, J. (2010) 'Virtualizing high performance computing', *SIGOPS Operating Systems Review*, Vol. 44, pp.136–145.

Sousa, E., Maciel, P., Medeiros, E., Souza, D., Lins, F. and Tavares, E. (2012) 'Evaluating eucalyptus virtual machine instance types: a study considering distinct workload demand', *3th International Conference on Cloud Computing, GRIDs, and Virtualization*, Nice, France, pp.130–135.

Turner, D., Oline, A., Chen, X. and Benjegerdes, T. (2003) 'Integrating new capabilities into NetPIPE', *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Springer, Berlin, Heidelberg, pp.37–44.

Vogelsberger, M., Genel, S., Springel, V., Torrey, P., Sijacki, D., Xu, D., Snyder, G.F., Bird, S., Nelson, D. and Hernquist, L. (2014) 'Properties of galaxies reproduced by a hydrodynamic simulation', *Nature*, Vol. 509, No. 7499, pp.177–182.

Xavier, M.G., Neves, M.V., Rossi, F.D., Ferreto, T.C., Lange, T. and De Rose, C.A. (2013) 'Performance evaluation of container-based virtualization for high performance computing environments', *21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, IEEE, Belfast, UK, pp.233–240.

Ye, K., Jiang, X., Chen, S., Huang, D. and Wang, B. (2010) 'Analyzing and modeling the performance in Xen-based virtual cluster environment', *12th IEEE International Conference on High Performance Computing and Communications (HPCC)*, IEEE, Melbourne, Australia, pp.273–280.

Younge, A.J., Henschel, R., Brown, J.T., Von Laszewski, G., Qiu, J.F. and Geoffrey, C. (2011) 'Analysis of virtualization technologies for high performance computing environments', *4th IEEE International Conference on Cloud Computing*, IEEE, Washington, DC, USA, pp.9–16.