Performance Analysis of LXC for HPC Environments

Conference Paper · July 2015

DOI: 10.1109/CISIS.2015.53

6 authors, including:

Edward David Moreno Ordonez
Universidade Federal de Sergipe
115 PUBLICATIONS **108** CITATIONS
SEE PROFILE

Patricia Takako Endo
University of Pernambuco
27 PUBLICATIONS **271** CITATIONS
SEE PROFILE

Djamel Sadok
Federal University of Pernambuco
242 PUBLICATIONS **1,303** CITATIONS
SEE PROFILE

Stenio Fernandes
Federal University of Pernambuco
100 PUBLICATIONS **673** CITATIONS
SEE PROFILE

Performance Analysis of LXC for HPC Environments

David Beserra
and Edward David Moreno
Federal University of Sergipe
Aracaju, Sergipe, Brazil
dw.beserra@gmail.com

Patricia Takako Endo
University of Pernambuco
Caruaru, Pernambuco, Brazil
patricia.endo@upe.br

Jymmy Barreto, Djamel Sadok
and Stênio Fernandes
Federal University of Pernambuco
Recife, Pernambuco, Brazil
{jpsb, jamel, sflf}@cin.ufpe.br

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

***Keywords*—Cloud Computing, HPC, Container-based virtualization, Performance evaluation.**

I. Introduction

High Performance Computing (HPC) is a generic term for applications that make intensive use of one or more computational resources and are technical-scientific nature [1] [2], like simulations of large-scale structures, such as the Universe [3]. Currently, we can host many HPC applications on Cloud infrastructures; including Cloud Computing solutions for scientific HPC applications, such as Nimbus [4] and Neblina [5]. HPC applications have used Cloud Computing infrastructure since the advent of scientific clouds [4] and virtualized computer clusters [6] due 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) customization, elasticity, and cost reduction of resource maintenance [7], [6].

In Cloud Computing environment, **virtualization** is a key technology for enabling its operation and, despite advantages, virtualization commonly suffers performance penalties due to resource sharing; presenting worst performance than non-virtualized scenarios [8]. In this way, evaluate virtualization performance under HPC applications is relevant, specially when we consider Message Passing Interface (MPI) applications running on Beowulf Clusters [7].

Trying to overcome this performance issue, **container-based virtualization** solutions (such as Linux-VServer [1], OpenVZ [2] and Linux Containers (LXC) [3]) have been proposed by academia, and according to [9], *this type of virtualization offers a lightweight virtualization layer, which promises a near-native performance*.

Our investigation aims to answer which scenarios LXC can offer a better performance than hypervisor-based virtualization or even equal to native environments for HPC applications. 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 II presents virtualization basic concepts needed to understand this work; Section III describes the related works; Section IV presents our main goals and the methodology used to perform our experiments; we present and discuss the results in Section V; and finally, we describe our considerations and delineate future works in Section VI.

II. Virtualization Background

Currently, we have four approaches to provide resource virtualization: full virtualization, hardware-assisted virtualization, paravirtualization, and operating-system-level (OS-level) virtualization.

The **full virtualization** allows a guest OS running without kernel modifications. However, it imposes high overhead in VM performance. This overhead can be mitigated using **hardware-assisted virtualization**, a set of specific instructions to virtualization that is present in almost processors and in some I/O elements [10].

When using **paravirtualization** approach, hardware devices are accessed through special paravirtualized drivers, obtaining a better performance when compared to full virtualization, even when it is assisted by hardware [7]. However, the guest OS kernel must be modified in order to provide new system calls. This modification increases the performance because reduces the CPU consumption, but at same time, it reduces the security and increases the management difficulty.

The **OS-level virtualization** 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 another containers and the host OS [9].

[1] http://linux-vserver.org/
[2] http://openvz.org/
[3] https://linuxcontainers.org/

In this work, our main objective is to analyse OS-level virtualization performance against another virtualization technique. The virtualization solutions used in this work are presented in more detail in next Sub-section and the justifications for our choice are presented in Section III.

A. KVM and LXC

Kernel based Virtual Machine (KVM) [4] is an open source virtualization tool integrated to Linux kernel. This virtualization approach takes advantage of the Linux kernel development as own evolution [10]. 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).

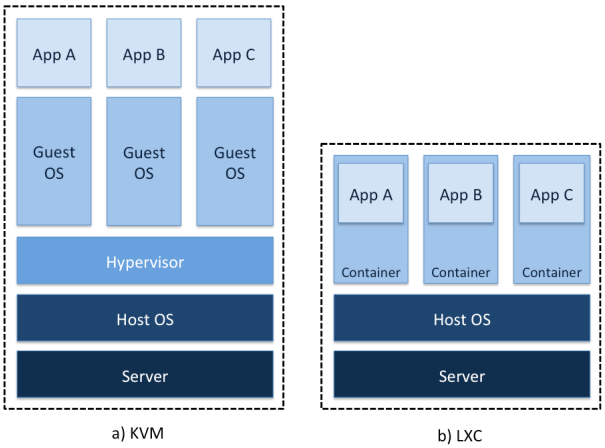

Fig. 1. KVM and LXC

Linux Containers (LXC) is a lightweight virtualization 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 (see Figure 1.b). The container shares the kernel with the host OS, so its 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 [11].

III. Related Work

Some works have already evaluated virtualization tools for HPC, such as [10] that compares KVM and Xen taking into consideration metrics like CPU and network performance, as well as the CPU consumption by VMs during tests. In regarding to network performance, authors considered a common scenario in Cloud Computing environments, in which VMs share the same host and send a great number of packets. On this scenario, KVM presented aggregated bandwidth better than Xen because KVM network emulation has a better host CPU utilization when processing data.

The work presented in [12] analysed the performance of open-source hypervisors - Xen, KVM, and VirtualBox - running some benchmarks in virtual clusters. They formulated a hypervisor ranking for HPC, and concluded 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.

However, these works did not considered OS-level virtualization technologies based on containers. Containers can offer advantages for HPC, such as a short bootstrap time, and a less disk space usage. Authors in [13] 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.

In other hand, authors in [14] compared the performance of OpenVZ, Xen, and KVM using VMs as web servers. OpenVZ presented results better than native in some situations; however the KVM paravirtualization instructions were not activated. This configuration could collaborate to KVM inferior performance, since KVM with paravirtualization instructions can obtain performance better than Xen [10], [12].

Authors in [9] innovates when analysing the performance of main OS-level virtualization 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 solution presenting performance close to a native environment. The network performance analysis considered bandwidth and delay according to packet size and their results indicated that LXC and VServer have better performance, with smaller delay and bigger bandwidth for any size of packet. *Since the HPC applications were tested, thus far, LXC demonstrates to be the most suitable of the container-based systems for HPC* [9].

For our work, we decided to use **KVM** because it has the best performance and is the best hypervisor-based virtualizaton solution according to literature [12], [15], and **LXC** because it can become the facto standard to system containerization, with the possibility to converge with OpenVZ to compose a same tool [16]. We extend the work done by [10], verifying how resource sharing among multiple VMs can impact 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 [9] and, for all experiment results, we take in consideration statistics analysis (such as variation and standard deviation) - aspect that is not contemplated in [9]. Additionally, different of all previous works, we also evaluate the intranode 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.

IV. Experimental Design

In this Section, we describe the experimental design used to evaluate both virtualization technologies. We adopted the methodology used by [17] to execute our measurements based on 5 main activities (Figure 2): firstly, do an analysis of LXC

[4] http://www.linux-kvm.org/page/Main_Page

and KVM architectures, implementations, and configuration options; the second activity is the planning of measurement that defines 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; the measurement activity is explained in Figure 3; the statistical treatment activity applies statistical methods to provide accurate information about our experiments.

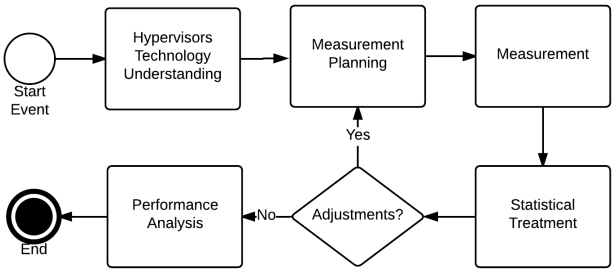

Fig. 2. Measurement methodology (adapted from [17])

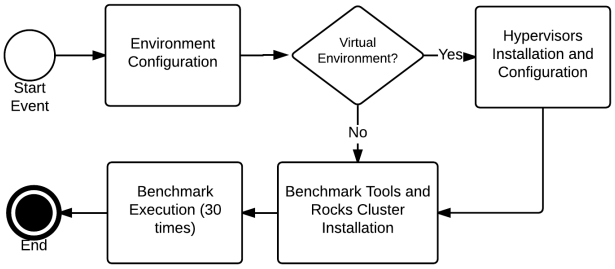

Fig. 3. Measurement activities

A. Infrastructure

Experiments were executed in 2 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 virtualization: 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.

B. Metrics and Benchmarking Tools

We choose analyse the performance of classical HPC performance metrics: CPU and communicating capacity (intranode and network-based communications). For each metric, we apply a specific benchmark, described in next sub-sections.

1) CPU Performance and HPL tool: To analyse the processor performance, we use the **High Performance Linpack (HPL)** benchmark [5]. HPL measures float point operations per second (Flops) done by a computational system during a linear equations system resolution [12]. 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 [6].

2) Communicating and NetPIPE: The main goal when analysing the inter-proccess comunicating performance is to understand how the communication bandwidth and delay are impacted by the additional virtualization layer. For this, we choose the **Network Protocol Independent Performance Evaluator (NetPIPE)** [6] 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.

C. Evaluated Scenarios

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

1) Determine the overhead caused by virtualization on CPU-bound applications performance; and
2) Determine how virtualization affects the inter-process communication performance, considering:
   a) Communications using only intranode comunication mechanisms;
   b) Communications 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 value were excluded as outliers; with the others 30, we computed the media and standard deviation. Next, we describe each environment detail according to these above goals.

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 1 physical node with 4 processor cores and 3096MB of memory; the virtual-1 had 1 VM with 4 cores and 3096MB of memory; the virtual-2 had 2 VMs with 2 cores per VM and 1536MB of memory per VM; and the virtual-4 had 4 VMs with 1 core per VM and 768MB of memory per VM.

TABLE I. 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 goals targeted, we evaluated the environments described in Table I, taking into consideration two possible execution scenarios:

[5]http://www.netlib.org/benchmark/hpl/

[6]http://bitspjoule.org/netpipe/

1) There is no communication between benchmark processes in execution;
2) The processes communicate with each other in a cooperative way.

1) Virtualization Overheads on CPU-bound Applications Performance: To evaluate the CPU performance when there is no interprocess communication, we instantiated 1 copy of HPL benchmark for each available processor. For instance, in the Native environment, we instantiated 4 HPL copies, each one executing in a distinct processor; while in the Virtual-4 environment, each VM had 1 HPL copy. So, as we had 4 VMs, we also had 4 HPL copies running. Here, we observed the aggregated performance (the sum of all individual HPL copies). On the other hand, when we have working in a cooperative way, using explicity interprocess communications; we ran only 1 HPL instance, using all available processors. The VMs in Virtual-2 and Virtual-4 environments were grouped in a cluster, with 1 HPL copy in execution.

2) Virtualization Overheads on Communication Performance: We used NetPIPE to verify the ocurrence of virtualization overheads in intranode interprocesses communications. First, we ran NetPIPE in Native and Virtual-1 environments; next, we ran NetPIPE in pairs of processes running in Virtual-2 environment, with each proccess in a distinct VM, to verify the virtualization effects in interprocesses 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. This experiment is relevant since the internal communication mechanisms can variate according to virtualization approach employed, and it can also influence in performance.

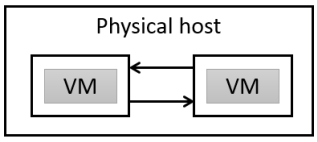

Fig. 4. VMs (or containers) communication inside the same physical host

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.To evaluate the network performance, we executed NetPIPE with processes allocated in different physical servers, measuring the performance according to packet size. To determine how the physical network virtualization affects real applications, we also ran the HPL benchmark in virtual clusters hosted in distinct servers.

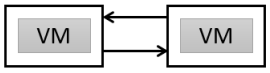

Fig. 5. Test setups using physical network

V. Results

This section presents the results - grouped according to our goals - we obtained from benchmark tests described in previous Section. For all metrics, we calculated media with standard deviation.

3) Virtualization Overheads on CPU-bound Applications Performance: In this sub-section, we present the results of HPL benchmark executed in a single server. Figure 6 shows results of CPU performance when executing HPL benchmark with no interprocess communication. We can observe LXC presented aggregated performance highest than KVM in Virtual-1 environment, and close to Native environment. Statistically, all variations were irrelevant (Low than 5%) for all environments and solutions.

When we divided physical resources in multiple virtual environments, both virtualization technologies presented a worse performance. This gradual decreasing affects KVM more poignantly because it 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 7 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 [9], [15]. 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 system is divided in more than one logic node, processes will communicate by using network protocols, increasing the processing and the communication latency.

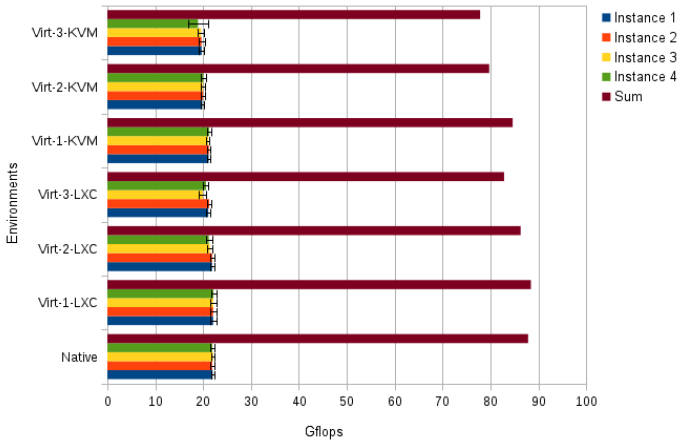

Fig. 6. HPL aggregated performance (same host)

4) Network Performance: Figure shows results of NetPIPE execution in a single server. We can observe that in Virtual-1 environment both virtualization technologies achieved communication bandwidth similar to achieved by Native environment, and the performance variations were insignificant. However,

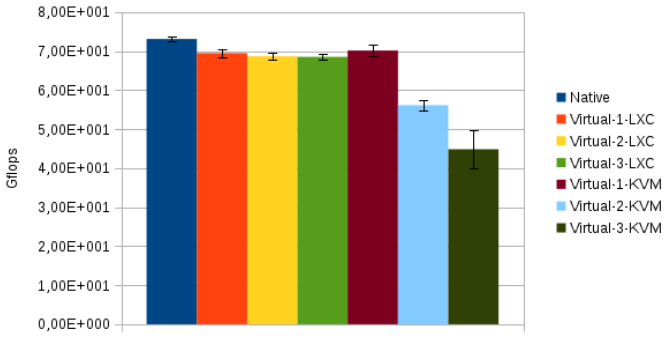

Fig. 7. HPL local clusters performance (same host)

when we divided the physical server in multiple virtual environments, the communication bandwidth achieved was reduced for both virtualization tools, especially KVM.

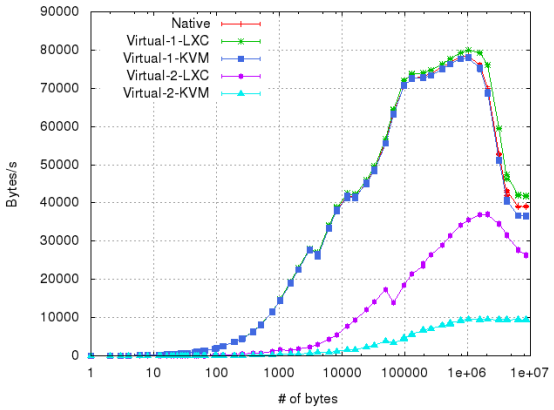

Fig. 8. Inter-process communication bandwidth (same host)

Since virtual-2 environment were configured with 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 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 ; we have highest values of delay when we have more requests to process. With exception of native, all environments presented considerable variations for both virtualization tools.

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

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

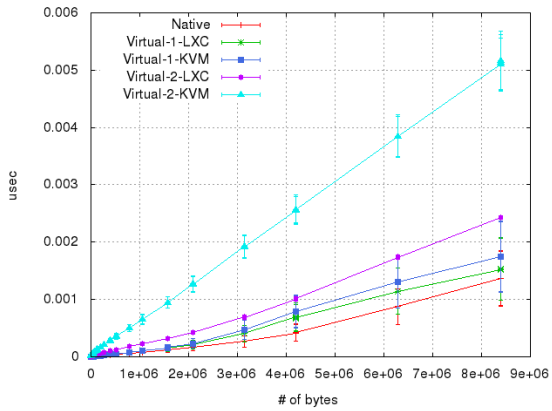

Fig. 9. Inter-process communication latency (same host)

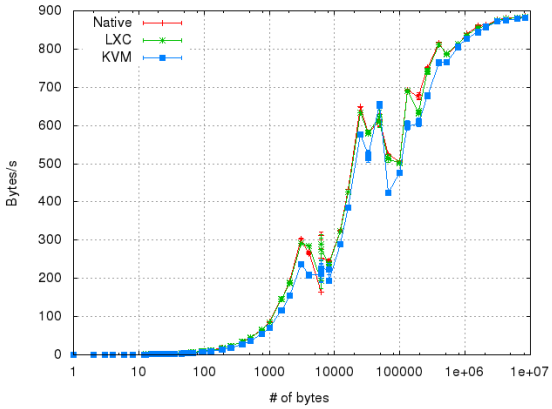

Fig. 10. Inter-process communication bandwidth (cluster)

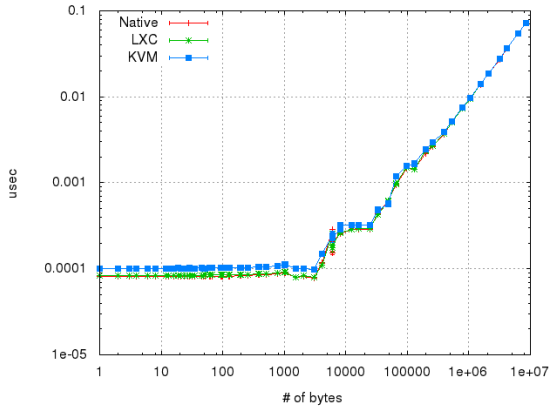

Fig. 11. Inter-process communication latency (cluster)

VI. Conclusion and Future Works

This paper presented a performance evaluation between two virtualization tools: KVM and LXC. According to our experiments, we observed LXC is more suitable for HPC than KVM. Considering a simple use case, 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 mul-

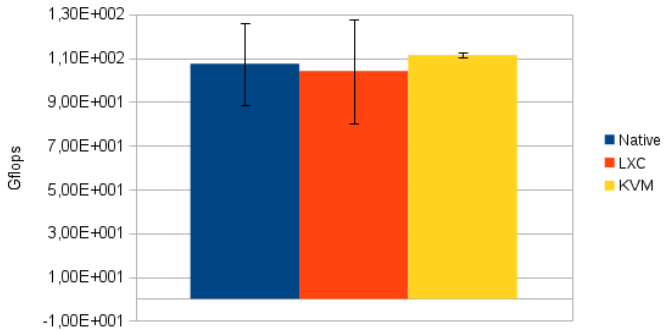

Fig. 12. HPL cluster performance (using network)

tiple and logic environments, both decrease their performance, 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 to variations on applications performance, LXC was better again, presenting less 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 an agreement penalty and, consequently, financial losses [6]

We can conclude in Cloud Computing environments, in which physical resources are splitted in multiple logic spaces, KVM does not present a good performance, and LXC can be considered the most suitable solution for HPC applications.

Despite container-based solution is a good alternative to overcome the overhead issues came from virtualization, during our experiments execution, we have faced to 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 [7]. Moreover, lxc-halt times-out and when we tried upgrade to "Jessie" breaks the container [8].

As future works, we plan to evaluate these virtualization technologies regarding to I/O performance for traditional Hard Drives, as well as for SSD devices, and to Graphical Processing Unit (GPU) for single server and cluster environments.

References

[1] Simons and Buell, "Virtualizing high performance computing," in *SIGOPS Operating Systems Review*. New York, NY, USA: ACM, 2010, pp. 136–145.

[2] D. Beserra, R. P. da Silva, K. Camboim, A. Borba, J. Araujo, and A. E. P. de Araujo, "Anlise do desempenho de sistemas operacionais hospedeiros de clusters virtualizados com o virtualbox," in *SBRC 2014 - WCGA*, Florianpolis - Brasil, may 2014.

[3] M. Vogelsberger, S. Genel, V. Springel, P. Torrey, D. Sijacki, D. Xu, G. F. Snyder, S. Bird, D. Nelson, and L. Hernquist, "Properties of galaxies reproduced by a hydrodynamic simulation," 2014, cite arxiv:1405.1418Comment: 32 pages, 5 figures, Nature Article (May 8, 2014).

[4] J. L. K. Keahey, T. Freeman and D. Olson, "Virtual workspaces for scientific applications," *Journal of Physics: Conference Series*, 2007.

[5] B. S. J. Fernandes, J. Barbosa and A. Mury, "Integration of cloud services in support to tests, simulations and knowledge dissemination with cluster environments," *Computer Systems (WSCAD-SSC)*, pp. 72–79, 2012.

[6] J. Napper and P. Bientinesi, "Can cloud computing reach the top500?" in *Proceedings of the Combined Workshops on UnConventional High Performance Computing Workshop Plus Memory Access Workshop*, ser. UCHPC-MAW '09. New York, NY, USA: ACM, 2009, pp. 17–20.

[7] K. Ye, X. Jiang, S. Chen, D. Huang, and B. Wang, "Analyzing and modeling the performance in xen-based virtual cluster environment," in *Proceedings of the 2010 IEEE 12th International Conference on High Performance Computing and Communications*, ser. HPCC '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 273–280.

[8] N. Regola and J.-C. Ducom, "Recommendations for virtualization technologies in high performance computing," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, Nov 2010, pp. 409–416.

[9] M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, and C. A. De Rose, "Performance evaluation of container-based virtualization for high performance computing environments," in *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*. IEEE, 2013, pp. 233–240.

[10] L. Nussbaum, F. Anhalt, O. Mornard, and J.-P. Gelas, "Linux-based virtualization for HPC clusters," in *Proceedings of the Linux Symposium*, Jul. 2009.

[11] Oracle. (2014) Oracle linux administrator's solutions guide for release 6 - chapter 9 linux container. [Online]. Available: http://goo.gl/qjiQR4

[12] A. J. Younge, R. Henschel, J. T. Brown, G. von Laszewski, J. Qiu, and G. C. Fox, "Analysis of virtualization technologies for high performance computing environments," in *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing*, ser. CLOUD '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 9–16.

[13] K.-T. Seo, H.-S. Hwang, I.-Y. Moon, O.-Y. Kwon, and B.-J. Kim, "Performance comparison analysis of linux container and virtual machine for building cloud," *SAdvanced Science and Technology Letters Vol.66 (Networking and Communication 2014)*, pp. 105–107, 2014.

[14] J. Che, Y. Yu, C. Shi, and W. Lin, "A synthetical performance evaluation of openvz, xen and kvm," in *Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific*. IEEE, 2010, pp. 587–594.

[15] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," *IBM technical report RC25482 (AUS1407-001), Computer Science*, 2014.

[16] R. Dua, A. R. Raja, and D. Kakadia, "Virtualization vs containerization to support paas," in *Cloud Engineering (IC2E), 2014 IEEE International Conference on*. IEEE, 2014, pp. 610–614.

[17] E. Sousa, P. Maciel, E. Medeiros, D. Souza, F. Lins, and E. Tavares, "Evaluating eucalyptus virtual machine instance types: A study considering distinct workload demand," in *CLOUD COMPUTING 2012, The Third International Conference on Cloud Computing, GRIDs, and Virtualization*, 2012, pp. 130–135.

[7] https://wiki.debian.org/LXC
[8] http://lwn.net/Articles/588309/