Improved isolation process management of LXC-Linux container under high performance computing

[1]Anju Lalwani,[2] Dr. Dhaval Vyas
[1]Center Manager,[2]Dean(MPhil), C U Shah University
[1]Academics,
[1]Thinkzzy Innovative Solutions PVT LTD, Ahmedabad, India.

Abstract : Not much research work is done in container technology for HPC. Virtualization technologies for HPC have performance overhead whereas container technologies for HPC is light weight, occupy fewer resources and have less performance overhead in comparison with virtualization. If container isolation performance gets improved more advantages of HPC technologies can be taken. Many container technologies are available for HPC from them LXC-containers are suitable most. LXC had an issue in limiting resources a container can use, which is now possible to limit. One of them is limiting the number of processes. This paper discusses that possibility.

IndexTerms **-** High performance computing (HPC), LXC, Linux Container, Isolation Performance, Process Management**.**

I. INTRODUCTION

HPC

High Performance Computing

HPC has the ability to hold and explore enormous amounts of data at high speed. Jobs that can take long time using normal computers can be done in days or even minutes. It can be used to form and work out extremely complex problems across a variety of high worth sectors.

LXC

LXC (Linux Containers) is an operating-system-level virtualization way for management several isolated Linux systems (containers) on a control host using a solo Linux kernel.

The Linux kernel provides the cgroups facilit that allows constraint and prioritization of resources (CPU, memory, block I/O, network, etc.) without the want for opening any virtual machines, and also namespace isolation facility that allows total isolation of an application's sight of the operating environment, including process trees, networking, user IDs and mounted file systems.

LXC combines the kernel's cgroups and maintain for isolated namespaces to offer an isolated tone for applications.

Initial Release 2008[34]

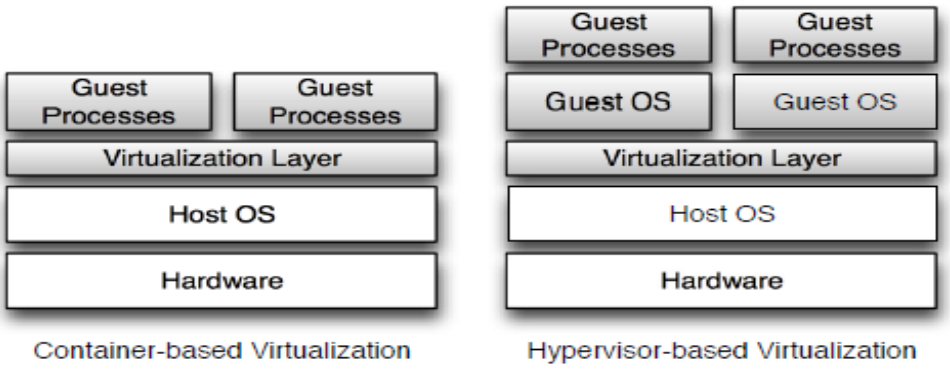

Figure 1. comparison of container-based and hypervisor-based virtualization [1]

LXD

LXD is system container manager, basically an alternative to LXC's tools, not a "rewrite of LXC, in fact it's building on top of LXC to provide a new, better user experience."[35]

Performance Isolation

Performance isolation between virtual machine (VMs) mean to the ability of separating the secular actions or limiting the secular intrusions of multiple VMs among each other, in spite of them running on the alike physical host and sharing physical resources such as processors, memory, and disks.

Why LXC

Mostly , security and isolation aspects of containers need an additional detailed examination.[32]

Hpc can use cloud infrastructure but because of several problems of virtualization overhead it was not considered. Container based virtualization has reduced this overhead and it's possible to have near-native performance.

LXC was not able to provide satisfactory isolation, it could not control guest from intruding host system.[2]

LXC is used in this work because it can be used widely as system containerization and compete with OpenVZ in making same tool.[2]

Mesos uses LXC to separate resources and isolate them from MPI(HPC software technology) and Hadoop frameworks.[36]
Lxc container has become very popular because it's by default available with linux operating system.
Container based virtualization systems, such as LXC, are foundation of the future cloud computing and have turned out the most admired resolution under Paas/Iaas cloud platforms especially after the launch and popularity of docker.[37]

Why HPC

HPC clusters, the Grid, hosting centers and PlanetLab need isolated and efficient system virtualization [33]

If hpc will be easily available on cloud, common man having high configuration requirement for business or for research can also make use of it with lowest investments and without getting involved in too much technical aspects.

Why Isolation Performance

Cloud providers have tackled challenges of how to attend growing demands in a scenario where the workloads within a virtual domain could not interfere the performance of other workloads running on the same physical hardware. This is a key motivation behind many works that explore performance isolation issues in many kinds of virtualization architectures
Cloud demand is growing and with its demand are its challenges growing and one main challenge is to give isolated performance where virtual area's performance is not interfered by other virtual area running on the same physical machine. Many works have been done to overcome this challenge in different virtualization architectures and works are still being done to achieve isolation performance fully and satisfactorily. [37]

Security and isolation features of containers need an extra careful study.[32]

Problem Explanation

The base of cloud computing is virtualization. Computer resources like processor, RAM, Storage, Network could be shared with more than one user remotely with the help of virtualization.

There are two types of virtualization available.
1) Hypervisor based virtualization
2) Container based virtualization

Earlier virtualization was possible through hypervisor based virtualization. In this type of virtualization each user is given separate operating system and through this operating system each user can run application separately and isolate from other users running on the same system.

This separate os for each user creates some overhead in performance especially in HPC.

Whereas the container based virtualization doesn't provide separate os for each user. It isolates resources through kernel namespaces and provides near-native computer performance.
Because of not giving separate os for each user isolation and security was weak but now container based virtualization has improved isolation and security slowly and gradually.

This paper discusses the same.

Because of less performance overhead in container based virtualization as compared to hypervisor based virtualization and improved isolation and security it is suggestible to use container based virtualization in HPC.[1]

LXC is widely used in HPC.[2]

As mentioned earlier, container based virtualization has improved isolation and security day by day. Earlier to limit number of processes was not possible. Because of which user could create unlimited processes and exhaust all memory. Which is called fork bomb and fork bomb crashes system.[1]

Following table shows that fork bomb test failed in LXC.

Table I
PERFORMANCE ISOLATION FOR LU APPLICATION. THE RESULTS REPRESENT HOW MUCH THE APPLICATION PERFORMANCE IS IMPACTED BY DIFFERENT STRESS TESTS IN ANOTHER VM/CONTAINER. DNR MEANS THAT APPLICATION WAS NOT ABLE TO RUN.

	LXC	OpenVZ	VServer	Xen
CPU Stress	0	0	0	0
Memory	88.2%	89.3%	20.6%	0.9%
Disk Stress	9%	39%	48.8%	0
Fork Bomb	DNR	0	0	0
Network Receiver	2.2%	4.5%	13.6%	0.9%
Network Sender	10.3%	35.4%	8.2%	0.3%

Figure 2. results of performance isolation for LU application[1]

But now it is possible to limit no. of processes. To limit no of processes lxc container uses cgroup. It uses pids cgroup to limit the no. of processes.[3]

Pids cgroup is available from Linux Kernel 4.3

Ubuntu operating system 19.10 eoan ermine has Linux kernel 5.3 and Pids cgroup is available in ubuntu 19.10.
On the time of experiment, ubuntu 18.10 and ubuntu 19.10 were available. Ubuntu 18.10 has Linux kernel 4.18 and ubuntu 19.10 has Linux kernel 5.3. Pids cgroup is available in Linux kernel 4.3 or greater i.e. Ubuntu 19.10 is selected for experiment.[4]

Why ubuntu

Ubuntu is one of the few (if not only) Linux distributions to come by default with everything that is needed for safe unprivileged Lxc containers .

Ubuntu 19.10 eoan ermine is available on aws marketplace[6]

Ubuntu 19.10 - Eoan

Ubuntu 19.10 - Eoan 20191114 | By Canonical Group, Ltd.
Linux/Unix, Ubuntu 19.10 Eoan | 64-bit (x86) Amazon Machine Image (AMI) | Updated: 4/1/20[6]

Ubuntu 19.10 comes with LXD 3.22 and LXC 3.22

Hardware explanation

Aws amazon ec2 provides variety of hardware resources as per user requirements.

If requirement is of high processing Ec2 provides compute optimized hardware/configuration

From the compute optimized c5 range is for HPC.

For this experiment c5.2xlarges is selected.

This configuration has 8 cpus,3.4 GHZ processor, Intel platinum 8124M, 21GB RAM, storage EBS only, network performance up to 25GB, 31Ecus[7]

Experiment Explanation

From root login create lxc container ubuntu 19.10 eaon. Start container service

You can see its configuration how much memory(RAM) initially it has

Initially it will have all memory of the main server.

When you create other container that will also show that it has all the main system memory

So each container can have all the memory available with main server.

Which can later create a problem if any one container occupies all memory other will starve and won't get service.

To avoid that, resource allocation must be controlled so that every user gets committed service or for the performance it paid for. It is possible through lxc config and lxc profile.

With the use of lxc config and lxc profile limits on computer resources can be set.
[3][5][8][9][11][12][15][16][17][27][28][29][30][31]

So by controlling number of processes, user has limited number of memory and fork bomb can be prevented.

This experiment worked well by controlling number of processes in container config cgroup. User can't exhaust system or crash system by creating unlimited number of processes.

As executing fork bomb is risky, before running fork bomb, factorial function was called. It is self calling function which creates many processes.

Factorial of 3 creates 3 processes. Number of processes was limited and factorial was executed to breach that limit.

Factorial function was not able to work beyond the number of processes allocated to container. Whenever limit was reached system used to give message "resource temporary unavailable".

After checking with factorial function, fork bomb was executed. Fork bomb calls itself unlimited times and tries to exhaust all memory but it was not able to run after allocated number of processes "resource temporary unavailable" message was displayed and system was not crashed as before it used to.

Conclusion

Container based technologies were not able to isolate resources properly before CPU was the only resource which could be isolated perfectly and LXC was not able to perform isolation in terms of memory,disk and network. Fork bomb used to crash system but now it is possible to limit number of processes and prevent fork bomb.
Container resources can be limited and give better performance and isolation in HPC i.e HPC can take advantage of LXC container based technologies.

Users or institutes having diverse software packages and configurations necessities use HPC clusters for those container technologies are helpful.

Mesos, which uses LXC to isolate resources from MPI and Hadoop frameworks, can now benefit from current resource limiting, securing and isolating facilities of LXC.

YARN (Yet another resource negotiator) also isolates resources of memory per application with the help of LXC and cgroups.[36]

REFERENCES

[1] Xavier, Miguel G., et al. "Performance evaluation of container-based virtualization for high performance computing environments." *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*. IEEE, 2013.

[2] Beserra, David, et al. "Performance analysis of LXC for HPC environments." *Complex, Intelligent, and Software Intensive Systems (CISIS), 2015 Ninth International Conference on*. IEEE, 2015.

[3] https://www.kernel.org/doc/Documentation/cgroup-v1/pids.txt

[4] https://releases.ubuntu.com/

[5] linuxcontainer.org/lxc/gettingstarted

[6] https://aws.amazon.com/marketplace

[7] https://aws.amazon.com/ec2/instance-types/

[8] https://www.linuxjournal.com/content/everything-you-need-know-about-linux-containers-part-i-linux-control-groups-and-process

[9] https://www.linuxjournal.com/content/everything-you-need-know-about-linux-containers-part-ii-working-linux-containers-lxc

[10] https://www.kernel.org/doc/html/latest/admin-guide/README.html

[11] https://www.kernel.org/doc/Documentation/cgroup-v2.txt

[12] https://www.kernel.org/doc/Documentation/cgroup-v1/memory.txt

[13] https://www.dell.com/downloads/global/products/pedge/en/server-poweredge-r610-tech-guidebook.pdf

[14] https://ubuntu.com/download/cloud

[15] https://linuxcontainers.org/pt_br/lxc/manpages//man5/lxc.container.conf.5.html

[16] https://linuxcontainers.org/lxd

[17] https://linuxcontainers.org/lxc/manpages/man5/lxc.system.conf.5.html#lbAE

[18] https://help.ubuntu.com/lts/serverguide/lxc.html

[19] https://help.ubuntu.com/community/EC2StartersGuide

[20] https://en.wikipedia.org/wiki/Xeon

[21] https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/terminating-instances.html

[22] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-security-groups.html

[23] https://discuss.linuxcontainers.org/t/comparing-lxd-vs-lxc/24

[24] https://console.aws.amazon.com/support/home#/case

[25] https://aws.amazon.com/training/

[26] https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc

[27] http://manpages.ubuntu.com/manpages/bionic/en/man1/lxc.config.set.1.html

[28] http://manpages.ubuntu.com/manpages/bionic/man3/YAML::Tiny.3pm.html

[29] http://manpages.ubuntu.com/manpages/bionic/man7/cgroups.7.html

[30] http://manpages.ubuntu.com/manpages/bionic/man7/lxc.7.html

[31] http://man7.org/linux/man-pages

[32] Morabito, R., Kjällman, J., & Komu, M. (2015, March). Hypervisors vs. lightweight virtualization: a performance comparison. In *Cloud Engineering (IC2E), 2015 IEEE International Conference on* (pp. 386-393). IEEE.

[33] Soltesz, Stephen, et al. "Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors." *ACM SIGOPS Operating Systems Review*. Vol. 41. No. 3. ACM, 2007.

[34] https://en.wikipedia.org/wiki/LXC

[35] *"Linux Containers - LXD - Introduction". linuxcontainers.org. Retrieved 2020-04-14.*

[36] Xavier, Miguel Gomes, Marcelo Veiga Neves, and Cesar Augusto Fonticielha De Rose. "A performance comparison of container-based virtualization systems for mapreduce clusters." *Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on*. IEEE, 2014.

[37] Xavier, Miguel G., et al. "A Performance Isolation Analysis of Disk-Intensive Workloads on Container-Based Clouds." *Parallel, Distributed and Network-Based Processing (PDP), 2015 23rd Euromicro International Conference on*. IEEE, 2015.