Reprodução de Avaliação de Desempenho em Sistemas de Virtualização

Pedro E. Melha Lemos[1]

[1]Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Faculdade de Informática -– Bacharelado em Ciência da Computação
Av. Ipiranga, 6681 – Bairro Partenon – CEP 90619-900 – Porto Alegre – RS – Brazil

pedro.elpidio@acad.pucrs.br

Abstract. *The virtualization adoption has increased on distributed systems environments, and recently it has gaining importance on high performance environments. Alongside the virtualization, due to the most known notion, there are performance losses. Thus it has been seeking a virtualization system which comes with all attributes of the most known virtualization, but with higher performance. As the container-based virtualization was getting popular, there were doubts at the container-based virtualization on reaching similar to native performance. On the article "Performance Evaluation of Container-based Virtualization for Highperformance Computing Environments" [XAVIER et al. 2013], it was seeking a performance evaluation of this kind of virtualization. This articles seeks to report the current progress of the referred article reproduction [XAVIER et al. 2013], shows computation performance evaluation and compare the performance between the native machine and a container.*

Resumo. *A adoção da virtualização tem crescido em ambientes de sistemas distribuídos, e atualmente vem ganhando importância em ambientes de alto desempenho. Junto com a virtualização, devido à noção mais conhecida, há perdas de dempenho. Buscava-se então um sistema de virtualização que possuísse todas as características da virtualização mais conhecida, porém com maior desempenho. Com a popularização da virtualização baseada em containers, duvidava-se da possibilidade de alcançar desempenho similar à máquina nativa. No artigo "Performance Evaluation of Container-based Virtualization for Highperformance Computing Environments" [XAVIER et al. 2013], procura-se avaliar o desempenho deste tipo de virtualização. Este artigo busca relatar o atual progresso da reprodução do artigo citado [XAVIER et al. 2013], apresentar avaliação de desempenho de computação e realizar a comparação de desempenho entre a máquina nativa, sistemas baseados em container e hypervisor.*

1. Introdução

Com os benefícios das técnicas de virtualização, e o suporte à mesma em processadores como Intel-VT [int] e AMD-V [amd], cresce a adoção de diferentes tecnologias para simulação de ambientes computacionais virtuais. No entanto, de acordo com XAVIER et al, em ambientes de computação de alto desempenho (HPC, do inglês: *High Performance Computing*) o seu uso havia sido evitado principalmente por causa da perda de

desempenho [XAVIER et al. 2013]. Inclusive, o mesmo autor menciona que até mesmo virtualização baseada em *hypervisor* possui perda de desempenho, especialmente em I/O.

Em contraponto, com a melhoria contínua de tecnologias de virtualização baseada em *containers*, torna-se possível chegar próximo ao desempenho nativo, devido à uma camada de virtualização leve o que a tornaria uma tecnologia poderosa para ambientes HPC [XAVIER et al. 2013]. Em XAVIER et al, a proposta é apresentar uma validação quanto ao desempenho de virtualização baseada em *containers* em ambientes de alto desempenho. Posto isso, este relatório tem por objetivo reproduzir o experimento presente em XAVIER et al [XAVIER et al. 2013] parcialmente, e verificar se a afirmação de que virtualização baseada em *containers* possui perda de desempenho mínima ainda é válida atualmente.

2. Virtualização

De acordo com XAVIER et al, o maior benefício da virtualização inclui a independência de *hardware*, disponibilidade, isolamento e segurança, sendo estes alguns dos fundamentos da computação em nuvem [XAVIER et al. 2013]. Inclusive, é usado em larga escala em servidores. No entanto, tecnologias de virtualização baseada em *hypervisor* como Xen [xen], VM-Ware [vmw] e KVM [kvm], possuem perdas de desempenho. Já as tecnologias de virtualização baseada em *containers*, como Linux-VServer [vse], OpenVZ [ope] e Linux Container (LXC) [lxc a], o desempenho é similar ao nativo.

2.1. Virtualização baseada em *hypervisor*

A virtualização baseada em *hypervisor* é considerada uma das mais populares. Em XAVIER et al é mencionado que a noção mais usual deste tipo de virtualização é que seja constituída de um monitor de máquina virtual (VMM, do inglês: *Virtual Machine Monitor*) no sistema operacional nativo que realiza toda uma abstração para que cada máquina virtual operante possa acessar os recursos de *hardware* da máquina hospedeira.

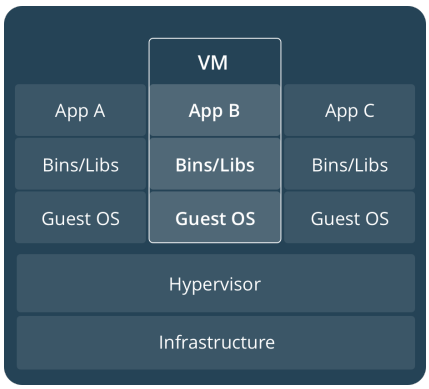

Figura 1. Diagram de virtualização baseada em *hypervisor* retirada de [doc]

É importante destacar que cada máquina virtual executa de forma isolada e possui sua própria instalação de sistema operacional [XAVIER et al. 2013]. Desta forma, se uma aplicação for destinada à um sistema operacional proprietário e o sistema operacional da máquina hospedeira for Linux, seria possível preparar uma máquina virtual com o sistema operacional necessário para tal aplicação.

2.2. Virtualização baseada em *containers*

Também conhecida como virtualização à nível de sistema operacional, de acordo com XAVIER et al, é particionado os recursos da máquina hospedeira visando permitir a criação de múltiplas instâncias de *user-space* [XAVIER et al. 2013]. O mesmo autor destaca que este tipo de virtualização opera na camada do sistema operacional, dispondo de uma abstração para que os processos do *container* executem como se fossem nativas. A Figura 2 ilustra este conceito.

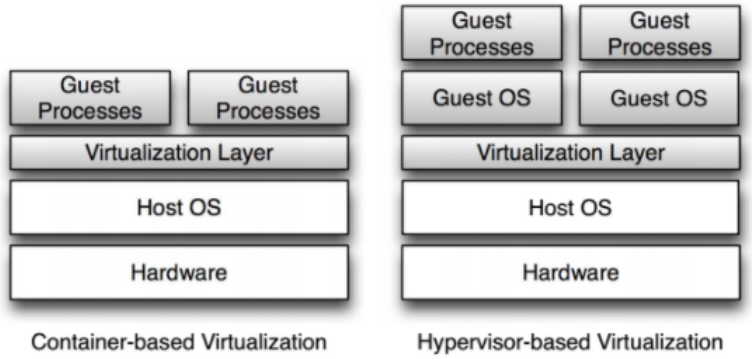

Figura 2. Comparação entre a virtualização baseada em *containers* e a baseada em *hypervisor* retirada de [XAVIER et al. 2013]

Para tanto, XAVIER et al afirma que na prática, a virtualização baseada em *containers*, funciona a nível de *system calls* e a baseada em *hypervisor*, a nível da abstração do *hardware*. Considerando que para a virtualização baseada em *containers* todas as instâncias compartilham o mesmo *kernel* do sistema operacional hospedeiro, isto não traria problemas de isolamento devido a cada *container* rodar como se fosse um sistema operacional independente, visto o isolamento ser feito através de *kernel namespaces* trazendo a ilusão ao *container* de ser o próprio sistema nativo [XAVIER et al. 2013].

Contudo, o controle de recursos por *container* é feito através de *Control Groups*; o que permite definir o nível de prioridade ao uso de processamento, armazenadamento, transferência e comunicação de acordo com XAVIER et al.

3. *Hardware* e ambiente de *software* utilizado

Semelhante ao apresentado em XAVIER et al, porém com somente um nodo, foi instalado o Ubuntu Server 16.04 LTS (Xenial Xerus) em uma máquina com dois processadores Intel Xeon E5520 em 2,27 GHz com 4 núcleos físicos cada em *hyperthreading* (disponibilizando assim 8 núcleos lógicos por processador), 8 MB de memória *cache* L3 [xeo] e 16 GB de memória RAM DDR3.

Foi utilizado o Linux *Containers* (LXC) em sua versão 1.0 disponível em PPA para o Ubuntu [lxc b] e o Docker 17.03.0-ce para realizar experimentos com a virtualização baseada em *containers*. Nos experimentos de virtualização baseada em *hypervisor* utilizou-se o KVM (QEMU emulator) 2.5.0. Por fim, também foi usado o Intel LINPACK Benchmark para realizar o teste de estresse no processador na versão 2017.2.015 [lin a].

4. Avaliação de desempenho de computação

Em XAVIER et al, as avaliações foram realizadas sobre o desempenho de *containers* do Linux-VServer, OpenVZ e LXC, e também, sobre a virtualização baseada em *hypervisor* do Xen. Por conta da instalação e configuração do Linux-VServer e do OpenVZ ter uma complexidade maior e consequentemente demandando mais tempo, foi somente reproduzido a avaliação de desempenho de *containers* no LXC e adicionalmente com o Docker. Além disso, foi reproduzido a avaliação de desempenho de sistemas baseados em *hypervisor*, mas com o KVM.

Para realizar a avaliação de desempenho de computação em um único nodo foi utilizado o LINPACK Benchmark em XAVIER et al. O objetivo deste *benchmark* é estimar o desempenho alcançado quando é feito o uso intensivo do processador. O LINPACK Benchmark é um programa que resolve sistemas de equações lineares [lin c] em uma matriz aleatória densa [BURKARDT].

4.1. Experimento original

Conforme relatado em XAVIER et al foi executado o *benchmark* do LINPACK em matrizes de ordem 3000 em todos os sistemas baseados em *containers* e foi feita uma comparação com Xen como apresentado na Figura 3. Todos os sistemas baseados em *containers* tiveram desempenho próximo ao nativo, chegando à ultrapassar 400 MFlops. Entretanto, o Xen não atingiu o mesmo desempenho, por ser um sistema baseado em *hypervisor*, havendo $4,3\%$ de perda no desempenho [XAVIER et al. 2013].

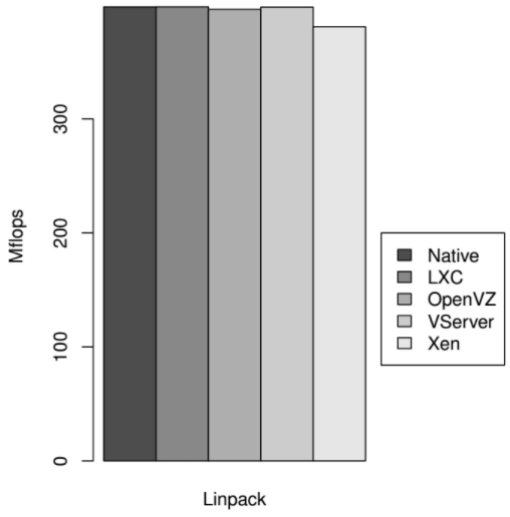

Figura 3. Comparação de desempenho de computação entre a máquina nativa, sistemas baseados em *containers* e *hypervisor* retirada de [XAVIER et al. 2013]

Acredita-se que por conta de existir somente um processo fazendo uso intensivo do processador permitiu que os sistemas baseados em *containers* atingissem desempenho similar ao nativo [XAVIER et al. 2013].

4.2. Reprodução de experimento

Após a resolução de alguns desafios no processo de instalação e preparação do ambiente para a realização dos experimentos, bem como a pesquisa realizada, tornou-se possível

preparar o ambiente e configurar o LINPACK para aproveitar os recursos da máquina [lin b] como, por exemplo, o total de núcleos físicos presentes na máquina e utilizando *hyperthreading*.

Em um primeiro momento, com a finalização da execução do LINPACK Benchmark, o desempenho aparentava estar instável executando na própria máquina nativa. Por outro lado, o mesmo *benchmark* executando em um *container* do LXC mostrava-se mais estável. Em média, a diferença era de $25,39\%$ entre o desempenho nativo e *container* do LXC como ilustrado na Figura 4.

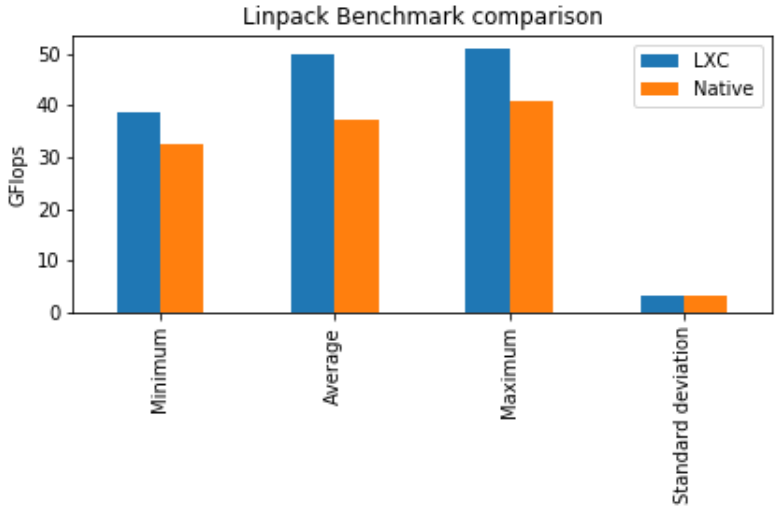

Figura 4. Análises a partir da comparação dos resultados do *benchmark* entre a máquina nativa e um *container* no LXC com matrizes de ordem 3000

Devido à tal instabilidade, foi constatado que não se estava aproveitando totalmente o processamento da máquina, e por conta do ano em que o experimento original [XAVIER et al. 2013] foi feito, tornou-se necessário aumentar a complexidade do problema. Portanto, a ordem da matriz para a realização do *benchmark* foi dobrada visando aumentar o estresse sobre o processador e foi re-excutado o *benchmark* cujo resultado está mais estável conforme a Figura 7.

Além disso, o mesmo *benchmark* foi executado em um *container* no Docker em uma máquina virtual no KVM. Como pode se observar na Figura 5, tanto o Docker quanto o LXC estão estáveis em 58 GFlops o que é semelhante ao desempenho nativo. Entretanto, a virtualização baseada em *hypervisor* do KVM apresentou instabilidades, mas houve picos que chegaram próximo ao desempenho nativo.

A fim de representar a condição de estabilidade de desempenho nativo e virtualizado (*containers* e *hypervisor*), tal como o mínimo e máximo de ambos, a Figura 6 contém o diagrama de caixa dos resultados obtidos do *benchmark*.

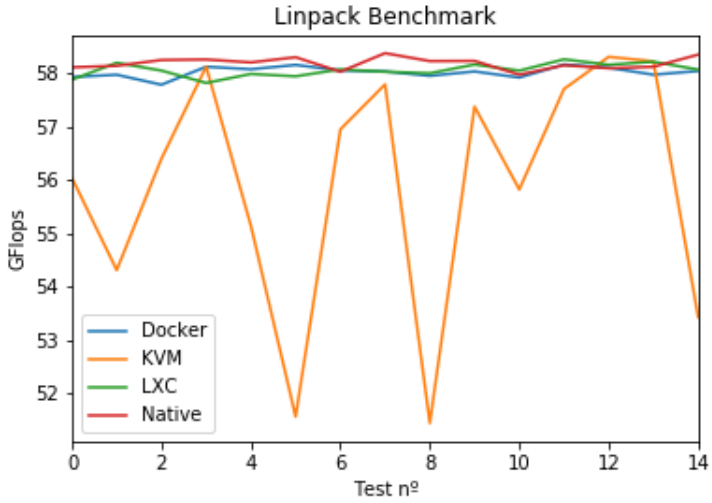

Figura 5. Resultados do *benchmark* obtidos na máquina nativa, em um *container* no LXC e no Docker, e uma máquina virtual no KVM com matrizes de ordem 6000.

Com o objetivo de extrair dados a partir destes resultados para análises, na Tabela 1 há informações detalhadas do desempenho mínimo, médio, máximo e desvio padrão nativo e virtualizado. Apesar de o desempenho entre o Docker e o LXC serem semelhantes, o LXC não só ficou mais estável, mas também obteve valores de mínimo, médio e máximo próximos ao nativo.

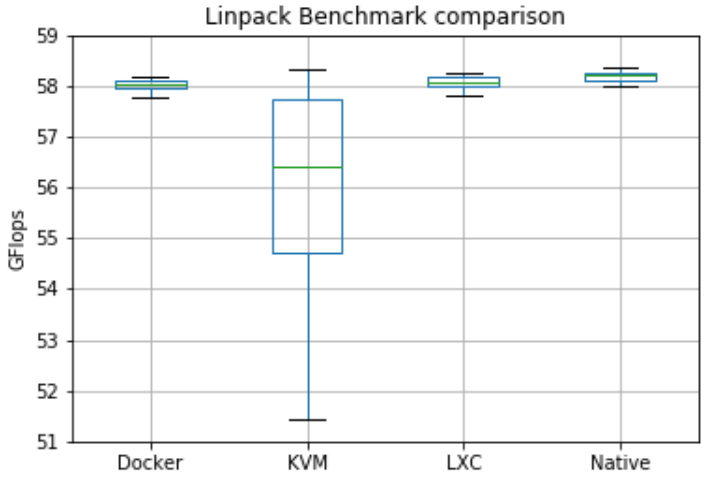

Figura 6. Diagrama de caixa sobre os resultados do *benchmark* obtidos na máquina nativa, em um *container* no LXC e no Docker, e uma máquina virtual no KVM com matrizes de ordem 6000.

O KVM teve desempenho mínimo de $51,44$ GFlops e uma média de $55,90$ GFlops, o que é justificável justamente por ser virtualização baseada em *hypervisor*. No entanto, supreendentemente o mesmo alcançou o máximo de $58,30$ GFlops que é muito próximo ao desempenho nativo. Contudo, o desvio padrão presente do desempenho na

máquina virtual do KVM foi consideravelmente alto.

	Docker	KVM	LXC	Native
Minimum	57, 784700	51, 448300	57, 817600	57, 972700
Average	58, 020260	55, 909073	58, 058647	58, 186693
Maximum	58, 163600	58, 307300	58, 259700	58, 375300
Standard deviation	0, 101485	2, 303103	0, 124846	0, 113420

Tabela 1. Dados obtidos a partir dos resultados do *benchmark* entre a máquina nativa, um *container* no LXC e outro no Docker, e uma máquina virtual no KVM com matrizes de ordem 6000.

Os dados presentes na Tabela 1 foram utilizados para gerar o gráfico presente na Figura 7, facilitando a comparação.

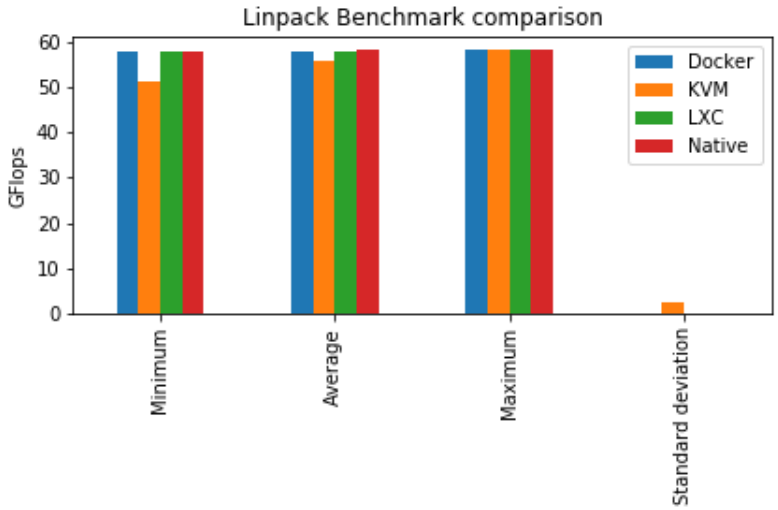

Figura 7. Análises a partir da comparação dos resultados do *benchmark* entre a máquina nativa, um *container* no LXC e outro no Docker, e uma máquina virtual no KVM com matrizes de ordem 6000.

5. Conclusão

Considerando que o *hardware* cuja as experiências foram feitas é similar ao apresentado em XAVIER et al, foi possível alcançar um ganho de aproximado de 10 *GFlops* (em torno de 20%) conforme reproduzido. Tendo em vista os avanços tecnológicos desde o ano em que os experimentos originais foram feitos [XAVIER et al. 2013], acredita-se que a razão desse ganho desempenho se deve às melhorias e constantes refinamentos nos últimos anos tanto no *kernel* Linux quanto nas tecnologias utilizadas justifiquem o aumento de desempenho.

A afirmação apresentada em XAVIER et al ainda é válida atualmente: a perda de desempenho em virtualização baseada em *containers* é mínima se comparado com o desempenho nativo como o apresentado.

6. Experimentos futuros

Há o objetivo de executar outros *benchmarks*, visando avaliar desempenho de memória, disco e rede para avaliar seus respectivos desempenhos nativamente, em um *container* e

em uma máquina virtual.

7. Agradecimentos

Agradecemos ao Laboratório de Alto Desempenho por disponibilizar o equipamento para aprendizado e realização dos experimentos ao longo do semestre, as orientações do coordenador Cesar A. F. De Rose, suporte e aulas da disciplina ministradas por Tiago C. Ferreto, instruções e apoio técnico de Miguel G. Xavier, revisão e melhorias no texto sugeridas por Julio Pires.

Referências

Amd virtualization solutions. Disponível em: http://www.amd.com/en-us/solutions/servers/virtualization.

Docker get started, part 1: Orientation and setup. Disponível em: https://docs.docker.com/get-started/.

Intel linpack benchmark. Disponível em: https://software.intel.com/en-us/articles/intel-mkl-benchmarks-suite.

Intel math kernel library. Disponível em: https://software.intel.com/en-us/forums/intel-math-kernel-library/topic/295302. Intel Developer Zone.

Intel virtualization technology. Disponível em: https://www.intel.com/content/www/us/en/virtualization/virtualization-technology/intel-virtualization-technology.html.

Intel xeon e5520. Disponível em: https://ark.intel.com/pt-br/products/40200/Intel-Xeon-Processor-E5520-8M-Cache-2_26-GHz-5_86-GTs-Intel-QPI.

Linpack benchmark. Disponível em: http://searchdatacenter.techtarget.com/definition/Linpack-benchmark. SearchDataCenter.

Linux containers. Disponível em: https://linuxcontainers.org/.

Linux containers getting started. Disponível em: https://linuxcontainers.org/lxc/getting-started/.

Linux kvm. Disponível em: https://www.linux-kvm.org/.

Linux-vserver. Disponível em: http://linux-vserver.org/.

Openvz. Disponível em: https://openvz.org/.

Vm-ware. Disponível em: http://www.vmware.com/.

Xen project. Disponível em: https://www.xenproject.org/.

BURKARDT, J. The linpack benchmark. Disponível em: https://people.sc.fsu.edu/~jburkardt/c_src/linpack_bench/linpack_bench.html. Florida State University.

XAVIER, M. G., NEVES, M. V., ROSSI, F. D., FERRETO, T. C., LANGE, T., and ROSE, C. A. F. D. (2013). Performance evaluation of container-based virtualization for high performance computing environments. In *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 233–240. IEEE.