Análise do Desempenho da Virtualização Leve para Ambientes com Edge Computing baseada em NFV

Leon Valentim Porto Trindade[1] **e Luís Henrique M. K. Costa**[1]

[1]Grupo de Teleinformática e Automação - GTA
Universidade Federal do Rio de Janeiro - UFRJ
Rio de Janeiro - RJ - Brasil

{leon,luish}@gta.ufrj.br

Abstract. *The Network Functions Virtualization (NFV) paradigm is changing the way network services are provided, allowing the same degree of versatility and agility as ever seen in cloud computing. However, it is possible to implement as virtual functions through traditional virtualization in hypervisors or in lightweight virtualization, constituting a compromise between flexibility and performance. Thus, this work compares the performance in environments that use virtualization technology with hypervisor to those that uses a container, for the purpose of verify the best for operational environments allied to the concept of Edge Computing in NFV. However, it is intended to gauge the ability for the technologies to be transparent to the user regarding performance. In order to do so, comparisons of CPU, memory, disk I / O, network and application were made, evidencing a low overhead of lightweight virtualization, which comes close to the native scenario in certain cases.*

Resumo. *O paradigma de Virtualização de Funções de Rede (NFV) está mudando a forma pela qual os serviços de rede são providos, permitindo o mesmo grau de versatilidade e agilidade já visto na computação na nuvem. Porém, é possível implementar as funções virtuais através da virtualização tradicional baseada em hipervisores ou da virtualização leve, constituindo um compromisso entre flexibilidade e desempenho. Assim, este trabalho compara o desempenho em ambientes que utilizam tecnologia de virtualização com hipervisor aos que utilizam contêineres, a fim de verificar a mais adequada para ambientes operacionais aliados ao conceito de Edge Computing baseada em NFV. Com isso, pretende-se aferir a capacidade dessas tecnologias serem transparentes ao usuário em relação ao desempenho. Para tanto, foram feitos benchmarkings de CPU, memória, Entrada/Saída (I/O) de disco, rede e aplicação, evidenciando a baixa sobrecarga da virtualização leve, que chega a se aproximar do cenário nativo em certos casos.*

1. Introdução

Os provedores de serviços de telecomunicações deparam-se constantemente com desafios de mercado e assim buscam reduzir custos, sejam em termos de *Capital Expenditure* (CAPEX) e de *Operational Expenditure* (OPEX). Além disso, buscam o aumento da agilidade de configuração dos serviços e otimização da utilização de recursos. A virtualização de funções de rede (NFV – *Network Functions Virtualization*) promete resolver esses desafios porque permite a diminuição do *time-to-market* de serviços de rede

nas operadoras, uma vez que o lançamento de serviços pode ser feito com menores investimentos. Além disso, permite uma maior flexibilidade de configuração, migração, automação e gerência.

Atualmente, a infraestrutura de redes das operadoras possui algumas características bem definidas: nós topologicamente fixos, algumas funções de rede com redundância dentro de *black boxes*, demora no lançamento de novos serviços pela operadora, novos serviços significam instalação de novos equipamentos, lógica de controle em cada dispositivo com difícil evolução. Sendo assim, características que contrastam com a flexibilidade, portabilidade e resiliência propostas pela virtualização.

Virtualizar funções de rede, testá-las e então certificá-las em diferentes plataformas é um desafio importante para os fabricantes de equipamentos de rede. Em muitos casos, eles oferecem plataformas de computação parceiras ou ecossistemas a fim de fornecer uma oferta integrada. A oferta de uma solução integrada faz com eles tenham que garantir que a *Virtual Network Function* (VNF) irá atuar como esperado, uma vez que isso será verificado para interoperabilidade e desempenho. Além disso, as operadoras de telecomunicações necessitam de um desempenho previsível das funções de rede virtuais para que elas sejam capazes de atender aos Acordos de Nível de Serviço (SLAs – *Service Level Agreements*).

Por outro lado, outro conceito importante é o de *Edge computing* que modifica o paradigma de topologia centralizada para descentralizada [Ismail et al. 2015], pela utilização de recursos de computação, rede e armazenamento que estão mais próximos ao usuário. A ideia é mover o conteúdo e o serviço para longe dos nós centrais, ou seja dos *datacenters* ou da nuvem, para as extremidades lógicas da rede. Idealmente, seja a um salto de distância do usuário, ou quanto mais perto a aplicação e o conteúdo, melhor a qualidade de experiência (QoE – *Quality of Experience*) do usuário. A eliminação de um ambiente completamente centralizado faz com que os gargalos e potenciais pontos de falha sejam minimizados.

A *Edge computing* busca a redução do tempo de resposta ou latência, fazendo cache ou descarregando o conteúdo nas pontas. As vantagens da utilização desse conceito aliadas aos benefícios do NFV, fazem com que novos serviços possam ser desenvolvidos e a avaliação da qualidade desses serviços seja necessária. Visando atender essas necessidades, o presente trabalho analisa o desempenho de diferentes funções de rede em uma plataforma x86. Uma vez que essas plataformas, por sua vez, podem utilizar diferentes tipos de virtualização, um benchmarking foi realizado a fim de avaliar os possíveis gargalos destas soluções e qual é mais apta para ser inserida na extremidade da rede.

Este artigo está organizado da seguinte forma. A Seção 2 apresenta os principais conceitos e diferentes tipos de virtualização. Além disso, apresenta o conceito de contêineres com diferentes ferramentas que utilizam essa tecnologia. A Seção 3 apresenta trabalhos relacionados comparando-os com o presente trabalho. A Seção 4 apresenta a metodologia utilizada na avaliação de desempenho das ferramentas de implementação de NFV. Já a Seção 5 apresenta os resultados experimentais da análise. Por fim, a Seção 6 conclui o trabalho e aponta direções futuras.

2. Virtualização

Soluções de virtualização compartilham o acesso ao hardware do computador permitindo que diferentes fatias virtuais sejam executadas sobre ele. Em um ambiente de datacenter legado, sem virtualização, um simples computador poderia somente executar um único sistema operacional por um certo tempo. Esse sistema era responsável por controlar todos os dispositivos de hardware do computador, como CPU, memória, disco rígido, placas de vídeo, placas de rede e outros periféricos. Com a virtualização, múltiplos usuários têm a ilusão do controle total sobre o hardware.

A virtualização é peça fundamental para a computação na nuvem e o NFV. Os principais benefícios incluem independência de hardware, isolamento, ambientes de usuário seguros e aumento da escalabilidade. Recentemente, observou-se um aumento do número de soluções e de tecnologias de virtualização: as chamadas soluções baseadas em hipervisores hoje concorrem com a virtualização baseada em contêineres que, em tese, possui menor sobrecarga computacional. Além disso, houve a introdução de novas técnicas híbridas que prometem combinar as vantagens das anteriores.

2.1. Virtualização Tradicional

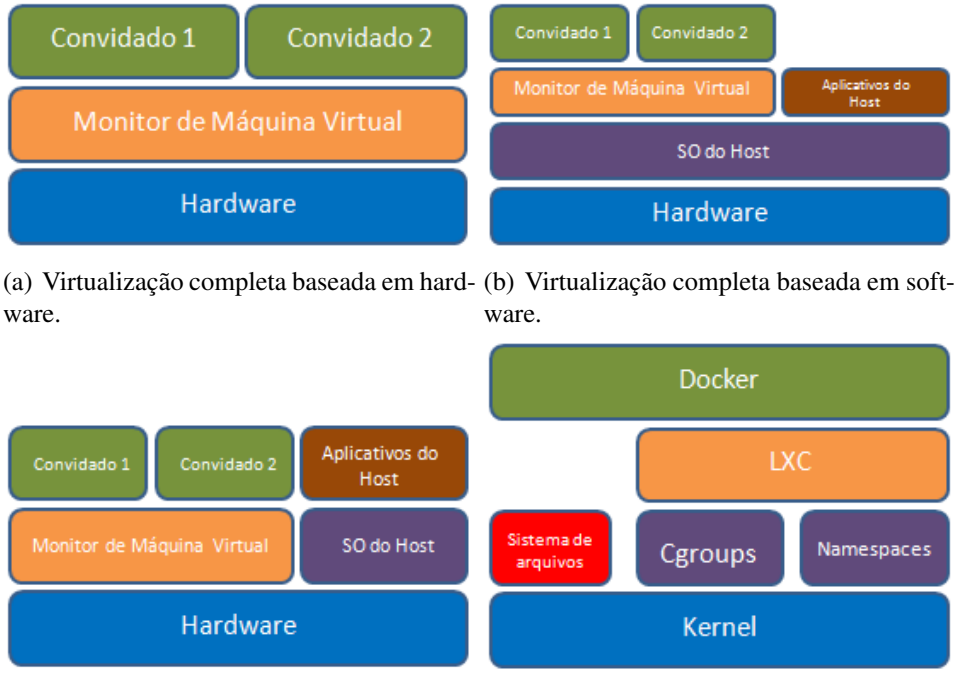
Uma solução de virtualização tradicional emprega uma camada de software chamada hipervisor, que se situa entre o hardware e as máquinas virtuais. O hipervisor controla o acesso aos hardware e realiza a alocação de recurso para cada máquina virtual (VM – *Virtual Machine*), fornecendo isolamento entre múltiplas VMs. Com isso, as VMs utilizam diferentes sistemas operacionais e hospedam suas próprias aplicações. Pode-se dizer que cada VM possui seu próprio ambiente computacional virtualizado, seu próprio sistema operacional e então sua própria abstração de hardware.

Na virtualização baseada em hardware, os hipervisores são executados diretamente no hardware, conhecidos como *bare-metal* (Figura 1(a)). Já na virtualização baseada em software (Figura 1(b)), o monitor de máquinas virtuais é executado sobre um sistema operacional hospedeiro, este sim executado diretamente sobre o hardware gerando mais uma camada de sobrecarga para o sistema. A virtualização híbrida é uma mistura das duas anteriores, ilustrada na Figura 1(c).

2.2. Virtualização Leve

A virtualização leve, ou virtualização baseada em contêineres, fornece um nível diferente de abstração em termos de virtualização e isolamento, quando comparada aos hipervisores. Conforme mencionado anteriormente, os hipervisores fornecem uma abstração de hardware, a qual resulta em uma sobrecarga em termos da própria virtualização do hardware e dos drivers virtuais do dispositivo. Isso significa que cada instância de VM é totalmente implementada com hardware virtual para suportar um sistema operacional visitante não modificado, executando sobre o hipervisor. Contêineres por outro lado fornecem um isolamento de processo a nível do sistema operacional, ou seja, para cada instância nova não é criada uma pilha completa de sistema operacional dedicado. Contêineres executam sobre o mesmo *kernel* do sistema operacional hospedeiro em um servidor físico, sendo que um ou mais processos podem executar dentro de cada contêiner.

A virtualização baseada em contêineres é considerada uma alternativa leve à virtualização baseada em hipervisores, uma vez que contêineres implementam isolamento



(a) Virtualização completa baseada em hardware.

(b) Virtualização completa baseada em software.

(c) Virtualização híbrida.

(d) Virtualização leve.

Figura 1. As diferentes arquiteturas de virtualização tradicional.

de processos a nível do sistema operacional da máquina hospedeira. Assim, evita-se a sobrecarga devido ao hardware virtualizado e aos drivers virtualizados de dispositivos. Um contêiner pode ser considerado um ambiente virtual pequeno e isolado, o qual inclui um conjunto de dependências específicas necessárias para executar determinada aplicação.

Os *namespaces* foram adicionados no *kernel* Linux na versão 2.6.24. São eles que permitem o isolamento de processos quando utiliza-se o Docker, além de serem os responsáveis por fazer com que cada contêiner possua seu próprio ambiente, ou seja, cada contêiner possui a sua árvore de processos e pontos de montagens no sistema de arquivos fazendo com que um contêiner não interfira na execução de outro.

Os *Control Groups* (cgroups) são os responsáveis por permitir a limitação da utilização de recursos da máquina hospedeira pelos contêineres. Com os *cgroups* é possível gerenciar a utilização de CPU, memória, dispositivos de entrada e saída (I/O – Input/Output) etc. Já os *namespaces* são responsáveis pelo isolamento de processos, gerenciamento de interfaces, gerenciamento de recursos de comunicação entre processos e sistemas de arquivos para montagem.

2.2.1. LXC

Algumas características como leveza e versatilidade tem facilitado o uso dos contêineres em diferentes contextos abrangendo desde computação na nuvem até cenários de Internet das Coisas (IoT – *Internet of Thing*), passando por NFV.

O LXC é uma ferramenta de criação de contêineres no núcleo do Linux. Os contêineres LXC são considerados um meio termo entre o *chroot* e uma máquina com vir-

tualização tradicional. O objetivo do LXC é criar um ambiente o mais próximo possível de uma distribuição Linux padrão sem precisar separar o *kernel* do sistema operacional hospedeiro. A sua arquitetura pode ser vista na Figura 1(d) , desconsiderando a camada superior adicionada pelo Docker [Docker 2013].

2.2.2. Docker

O Docker introduz um mecanismo de camadas subjacentes, junto com uma *Application Programming Interface* (API) funcional que permite facilmente criar, gerenciar e remover a aplicação dentro do contêiner. Devido à pequena sobrecarga, vários contêineres podem executar mesmo em dispositivos com recursos computacionais limitados, tais como plataformas *Single Board Computer* (SBC).

Além disso, é baseado no sistema operacional Linux e permite utilizar o estilo *Representational State Transfer* (REST) para gerenciar os contêineres através das suas imagens Docker. Um contêiner Docker é semelhante a uma máquina virtual, contudo o Docker é talvez mais eficiente que a VM, porque ele elimina o hipervisor e o sistema operacional visitante completamente, enquanto mantém os contêineres portáveis através do *kernel* do sistema operacional Linux.

O Docker foi desenvolvido com técnicas de *copy-on-write*. A ideia básica é que um novo recurso, seja ele um bloco de disco ou uma área de memória, só é alocado quando for modificado. O Docker usa um esquema de camadas (*layers*) montadas por meio da técnica *copy-on-write*, ou seja, um contêiner é basicamente uma pilha de camadas compostas por N camadas *read-only* e uma camada superior *read-write*.

3. Trabalhos Relacionados

A avaliação de desempenho em soluções virtualizadas está presente em diversos trabalhos na literatura com alguns deles abordando aplicações específicas como em [Eiras et al. 2016] que faz uma análise de desempenho de duas ferramentas de código aberto de virtualização, KVM e Docker, em um cenário com HTTP proxy. Adicionalmente, em [Varghese et al. 2016] foram apresentadas técnicas de benchmarkings leves na nuvem que são executadas rapidamente e podem ser utilizadas em tempo real.

Em [Al Jabry et al. 2014] é realizada uma investigação sobre os impactos de vários hipervisores no que diz respeito ao desempenho do sistema e com o objetivo de identificar qual é o hipervisor ótimo para satisfazer as necessidades do mercado. Os hipervisores VMWare, VirtualBox e VirtualPC foram comparados adotando um benchmarking de desempenho que analisa I/O de disco, memória, CPU e consumo de energia. Entretanto, o trabalho considera como cenário nativo um sistema operacional diferente do cenário virtualizado, podendo ter seus resultados mascarados devido à natureza das diferentes pilhas de sistemas operacionais hospedeiros. Já o presente trabalho leva em consideração o desempenho de rede, aplicação e utiliza o mesmo sistema operacional hospedeiro para todos os cenários avaliados trazendo mais confiabilidade às amostras.

Ademais, em [Pesic et al. 2016] foi feita uma análise de desempenho em 3 diferentes plataformas de armazenamento em sistemas de arquivos no Linux, que são ext4,

xfs e btrfs em hipervisores do tipo 2. Foram utilizadas as ferramentas de benckmarking Postmark e Bonnie++ para geração de testes no ambiente com ênfase na vazão de leitura e escrita sob diferentes condições de carga. Entretanto, a análise foi específica para disco que talvez não seja o aspecto mais importante em termos de desempenho de aplicações no ecossistema de NFV, uma vez que processamento e memória são cruciais e também são uma das contribuições do presente artigo.

Nos trabalhos [Mattos et al. 2012a] e [Mattos et al. 2012b], visando avaliar como serão os cenários e protocolos para Internet do Futuro, tendo como premissa que virtualização será a técnica principal para acomodar os elementos dessa nova Internet, os autores avaliaram o desempenho de 3 ferramentas de virtualização conhecidas que são Xen, VMWare e OpenVZ considerando a utilização para virtualização de roteador. Conduziram o experimento com ferramentas de benchmarking para medir a sobrecarga introduzida pela virtualização em termos de memória, processador, rede e desempenho de disco dos roteadores virtuais executando em servidores *Commercial off-the-shelf* (COTS).

Adicionalmente, avaliaram os efeitos de escalabilidade de máquinas virtuais no mecanismo de virtualização de rede Xen e os resultados mostraram que o Xen melhor atende os requisitos de roteadores virtuais, entretanto a análise está defasada devido a diversas melhorias nas ferramentas utilizadas, diferentemente do presente trabalho. Além disso, o artigo presente analisa explicitamente a virtualização assistida por hardware e a virtualização leve que é foco em ambientes operacionais nas principais empresas que lidam com aplicações na nuvem evidenciando um avanço no trabalho atual em relação aos citados acima.

Nenhum dos trabalhos anteriores mistura diversos tipos de virtualização e ferramentas atuais com análise de aplicações de missão crítica, ou seja, aquelas cruciais para a empresa como banco de dados, *Enterprise Resource Planning* (ERP), *Customer Relationship Management* (CRM) e plataformas de colaboração. Além disso, nenhum dos anteriores utiliza mais de uma ferramenta em uma única vertical de análise, como por exemplo é feito no caso de processamento para dupla validação do comportamento garantindo maior confiabilidade a análise. Ademais, o presente trabalho se preocupa com tratamento estatístico, uma vez que em todos os casos foram consideradas diversas iterações para cada ponto em análise.

4. Metodologia de Análise

Esta seção descreve a metodologia utilizada a fim de analisar o desempenho de diferentes tipos de virtualização na implementação de *Edge computing* baseada em NFV. Foram analisadas algumas das ferramentas de virtualização mais difundidas atualmente: *Kernel-based Virtual Machine* (KVM), VirtualBox, LXC e Docker. As ferramentas escolhidas representam diferentes tipos de virtualização, conforme explicado anteriormente. Uma das maiores dificuldades em avaliar ferramentas de virtualização é que algumas ferramentas comuns de benchmarking, têm suas medidas distorcidas pelo tempo interno ao ambiente virtualizado, porém essa dificuldade não prejudicou a confiabilidade da análise devido ao ajuste de *timestamp* e controle de *drift* temporal nos hipervisores utilizados.

Foram realizados benchmarkings que avaliam o desempenho de CPU, de memória, de entrada e saída de disco, de desempenho de rede, e de aplicações visando a verificação da qualidade de experiência do usuário, que está em foco nas operadoras de

telecomunicações em face da real ameaça das chamadas *Over-the-tops* (OTTs), ou seja, empresas com foco em conteúdo.

Para realizar a análise de desempenho, foi utilizado um cenário composto por um computador com processador Core i7, 8 GB de RAM e disco rígido de 1 TB. O computador executa um sistema operacional Linux, distribuição Ubuntu 16.4 com *kernel* versão 4.4.0-31 e placa de rede com velocidade de 100 Mbps. A configuração dos testes permitiu comparar o desempenho dos diferentes tipos de virtualização em 5 cenários distintos, ilustrados na Tabela 1. Ademais, a topologia de rede está ilustrada na Figura 2, com origem e destino de acordo com o cenário definido na Tabela 1.

Tabela 1. Cenários de análise.

Cenários	Plataformas
1º cenário	Nativo
2º cenário	Nativo + Docker
3º cenário	Nativo + LXC
4º cenário	VirtualBox + VM
5º cenário	KVM + VM

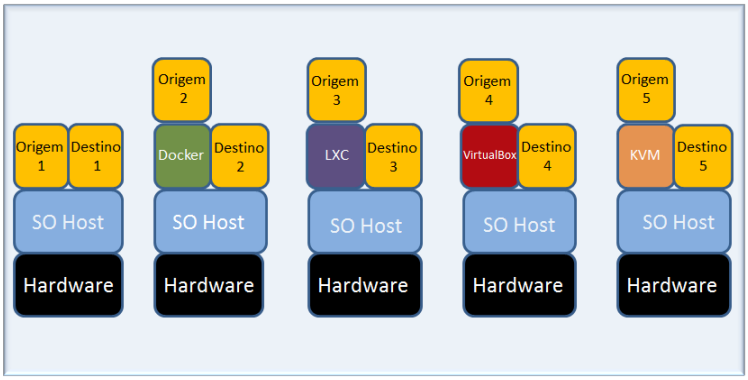


Figura 2. Cenários para avaliação do desempenho de rede.

5. Resultados Experimentais

Ferramentas de *benchmarking* sintético permitem gerar diferentes tipos de carga para avaliar o desempenho de algum subsistema específico do hardware. Assim, CPU, memória, I/O de disco, rede e qualidade da experiência do usuário são os principais componentes testados nesse trabalho, uma vez que eles são métricas importantes para ambientes com NFV. Para isso, as ferramentas utilizadas na avaliação foram o Y-cruncher [Yee 2009], o Sysbench [Kopytov 2004], o STREAM [McCalpin 1991], o Bonnie++ [Coker 2001] e o Iperf [McMahon 2003].

5.1. Desempenho de CPU

A fim de avaliar a sobrecarga da virtualização em termos do desempenho de CPU, foi utilizado o Y-cruncher, que é uma ferramenta de *benchmarking* com múltiplas *threads* desenvolvida para sistemas *multicore* que calcula o valor de π. Há outras aplicações para *stress* de CPU que realizam o cálculo de π, porém o Y-cruncher tem a vantagem de

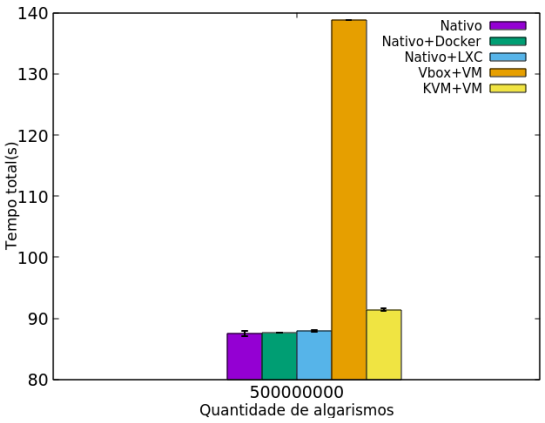


Figura 3. Desempenho de CPU: tempo de cálculo do número π com x algarismos utilizando o Y-cruncher.

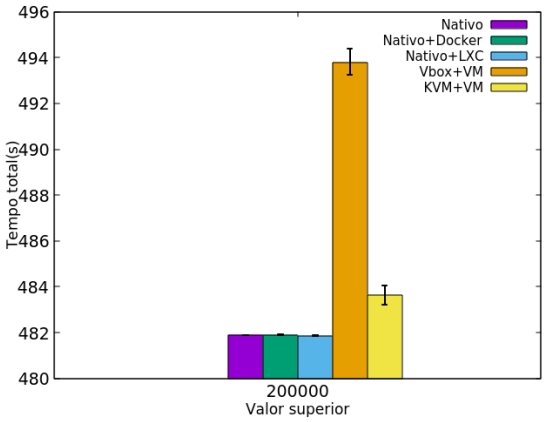


Figura 4. Desempenho de CPU: tempo de cálculo de N números primos até o valor superior x utilizando o Sysbench.

ser *multi-thread*. A métrica levada em consideração na ferramenta foi o tempo total de computação que consiste no tempo que é necessário para o processamento do resultado subtraído do tempo total de execução.

O valor tamanho de π considerado na ferramenta Y-cruncher foi 500 milhões de algarismos. Conforme observa-se na Figura 3, o desempenho da virtualização leve foi próxima à do ambiente nativo, entretanto o desempenho da virtualização baseada em software foi aquém das demais, evidenciando um gargalo de CPU na utilização do VirtualBox.

Os desempenhos obtidos tanto pelo Docker quanto pelo LXC aproximam-se do ambiente nativo devido a ambos compartilharem recursos do *kernel* hospedeiro utilizando o *Completely Fair Scheduler* (CFS) como escalonador de processos. Este, por sua vez, sempre busca normalizar as fatias de tempo para acesso ao processador em relação ao número total de tarefas para se aproximar do acesso multitarefa ideal ao processador, ou seja, executar os processos de maneira justa.

Ademais, os já mencionados cgroups e namespaces se encarregam do isolamento dos contêineres proporcionando a cada um deles um escalonamento de processos em grupo através do CFS que utiliza um mecanismo de árvore rubro-negra ordenada no tempo para construir uma linha de execução de tarefas futuras.

Apesar de também utilizar o CFS como escalonador de processos, o KVM obteve um desempenho inferior aos cenários com virtualização leve devido ao tratamento dado às interrupções pela sua arquitetura que depende da virtualização baseada em hardware e esta ainda possui uma sobrecarga nas trocas de contexto entre visitante e hospedeiro fazendo com que algumas instruções causem um número elevado ciclos de processamento.

Além do Y-cruncher, também foi utilizada a ferramenta Sysbench, que realiza o cálculo dos N números primos existentes até um determinado valor superior como critério de parada. O valor escolhido foi 200.000. Como pode ser visto na Figura 4, o desempenho do cenário que utiliza o Virtualbox foi bem inferior aos demais, devido à tentativa do sistema operacional visitante de executar código de anel 0 em anel 1 causando uma série

de falhas de instrução, uma vez que o código no anel 1 não possui permissão de executar instruções privilegiadas.

Em cada uma dessas falhas, o Virtualbox deve intervir emulando o código para obter o comportamento desejado e isso gera um sobrecarga elevada que degradada o desempenho nitidamente. Apesar de utilizar técnicas para melhoraria de desempenho como o componente chamado *Code Scanning and Analysis Manager* (CSAM) que desmembra o código executado pelo visitante e aciona o *Patch Manager* (PATM) que substitui esse código em tempo de execução, o desempenho do Virtuabox permaneceu abaixo dos demais.

5.2. Desempenho de Memória

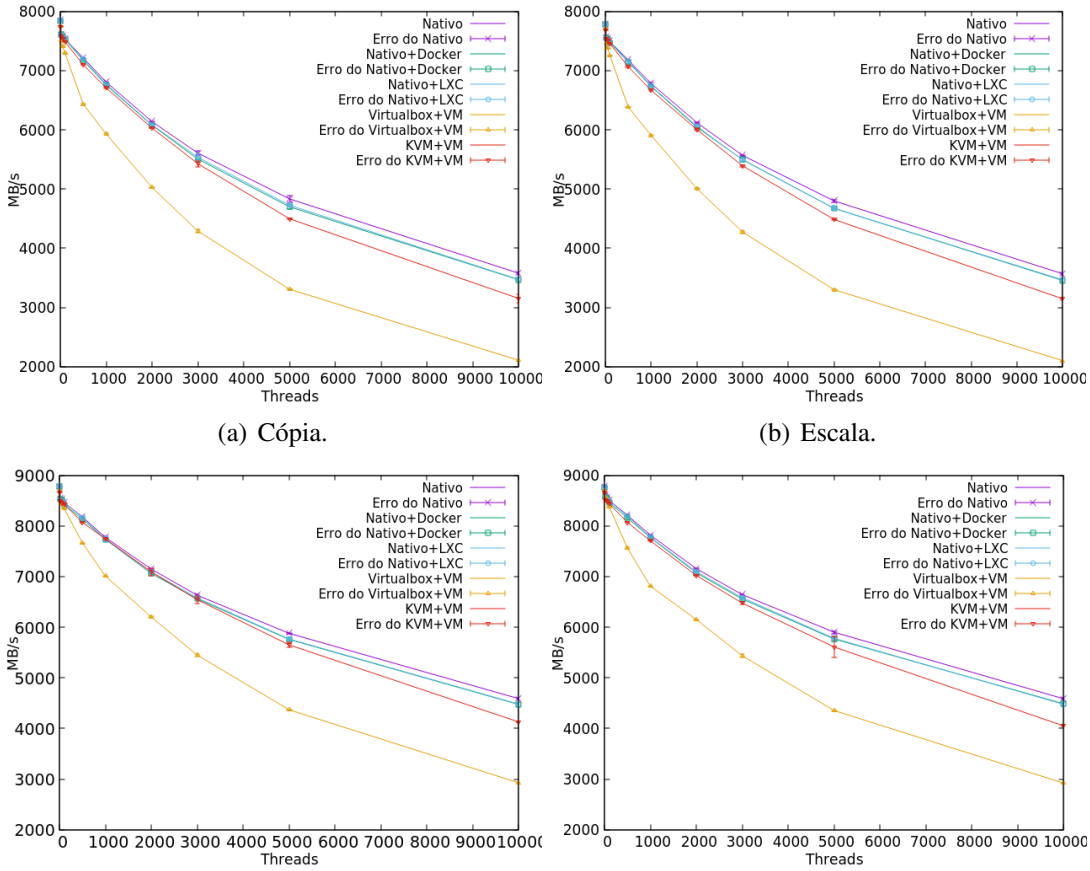
A fim de testar a velocidade de acesso memória, foi utilizada a ferramenta STREAM [McCalpin 1991]. Esta mede o desempenho da memória utilizando operações triviais de vetores no *kernel*, ou seja, gerando resultados para quatro operações diferentes sobre vetores. Conforme ilustrado na Figura 5, para as quatro operações, o desempenho do VirtualBox que representa a virtualização baseada em software é inferior aos demais, sendo aproximadamente 30% inferior a partir de 2.000 *threads*.

A tradução de endereços virtuais para físicos é uma operação que utiliza a memória intensamente, uma vez que o mapeamento deve ser feito de maneira hierárquica diversas vezes. Para reduzir essa sobrecarga, os processadores automaticamente guardam as traduções recentes em tabelas, chamadas *Translation Look-aside Buffer* (TLB). Com isso, a cada referência a memória, o processador checa antes o TLB para determinar se a tradução desejada já está em *cache*. Ademais, o sistema operacional atualiza o *Control Register 3* (CR3) durante uma mudança de contexto. Uma mudança no registrador CR3 estabelece um novo conjunto de traduções e então o processador invalida as entradas do TLB associadas ao contexto anterior.

O VirtualBox obteve um desempenho inferior ao demais devido à sua técnica de tradução de memória chamada *shadow pages*, a qual mantém uma tabela de blocos de memória de sombra derivada da tabela de memória do sistema operacional visitante. Quando o visitante está ativo, o hipervisor força o processador a utilizar essa tabela de sombra para realizar a tradução de endereços de memória e isso gera uma sobrecarga devido ao esforço de manter essa tabela de sombra válida através de rastreamento do estado da tabela de memória do visitante.

Os cenários com virtualização leve utilizam o cgroup de memória, o memcg, para alocar endereços físicos de memória, inclusive com seu próprio mapeamento de endereços mais utilizados recentemente (Least Recently Used - LRU) gerando um sobrecarga baixa e fazendo com que seu desempenho seja semelhante ao nativo.

Além disso, no cenário com KVM é construída uma tabela de endereços de memória para mapear os endereços físicos do visitante para os endereços físicos do hospedeiro como se o visitante acessasse seu próprio endereço físico. Essa técnica, conhecida pela Intel como *Extended Page Table* (EPT), faz com que o controle do mapeamento de memória fique no sistema operacional visitante fazendo com que o hipervisor não fique sempre rastreando o estado da tabela de memória do visitante para emular alguma alteração, reduzindo assim a sobrecarga como pode ser visto na Figura 5.



(a) Cópia. (b) Escala.

(c) Adição. (d) Tríade.

Figura 5. Desempenho de Memória: Vazão de operações em memória utilizando diferentes operações aritméticas com a ferramenta STREAM.

5.3. Desempenho de Disco

Também foi avaliado o desempenho de acesso ao disco rígido, para isso foi medida a sobrecarga de virtualização do disco nas tarefas de escrita e leitura, comparando a sobrecarga produzida por cada tecnologia de virtualização. Foi utilizada a ferramenta Bonnie++, um software livre de teste de disco que simula operações como criação, leitura e exclusão de pequenos arquivos. O Bonnie++ também testa o desempenho de acesso a diferentes regiões do disco rígido, acessando os setores no começo, meio e fim do disco.

As métricas levadas em consideração para a análise foram a velocidade de escrita representada como entrada e de leitura representada como saída do disco. Além disso, foi avaliada a quantidade de buscas aleatórias feitas com sucesso no disco, através da criação de 4 processos em paralelo pela ferramenta Bonnie++.

Como pode ser observado na Figura 6, tanto para 25, 50 e 100 arquivos, o pior desempenho é o da ferramenta Docker, a qual utiliza virtualização leve, frente às demais. A discrepância é grande até mesmo com o LXC que utiliza o mesmo tipo de virtualização. O Docker versão 1.12.6 executado no Ubuntu 16.04 utiliza o sistema de arquivos *AUFS* em conjunto com o ext4 como sistema de arquivos subjacente.

O *AUFS* agrupa diversas camadas em um único hospedeiro e as apresenta como

sendo uma única. Essas camadas são unificadas através de um processo que é chamado de *union mount*. Uma vez que o *AUFS* faz uma manipulação a nível de arquivos, basta uma pequena escrita em um arquivo que ele tem que ser copiado por inteiro, operação chamada de *copy up* gerando uma grande sobrecarga na escrita e leitura de arquivos conforme é observado na Figura 6.

Tanto o desempenho do Virtualbox quanto do KVM foram próximos ao cenário nativo. O primeiro emulou um controlador *Serial ATA* (SATA) no modo *Advanced Host Controller Interface* (AHCI) juntamente com uma imagem de disco com extensão *Virtual Disk Image* (VDI), já o segundo obteve um bom resultado devido a baixa sobrecarga inserida pelo KVM que manipula arquivos através de entidades chamadas *storage pools* e *volumes*. O *storage pool* padrão utilizado foi o baseado em diretórios com uma técnica de cache chamada *writeback* na qual estão habilitados para o sistema operacional visitante tanto o cache do hospedeiro para melhorar I/O de disco quanto o cache de operações de escrita no disco.

Para buscas aleatórias no disco feitas com sucesso após o Bonnie++ criar 4 processos filhos, o desempenho do Docker foi bom como é mostrado na Figura 7, devido as buscas não necessitarem de operações de cópia em diversas camadas, uma vez que basta achar a posição no disco para medir o *Input Output per Second* (IOPS). Por outro lado, tanto o desempenho do cenário com KVM quanto o do cenário com Virtualbox foram ruins devido as gargalos inseridos pelos sistemas de arquivos e controladores de I/O, no KVM através do `virtio-blk` e no VirtualBox com o SATA.

5.4. Desempenho de Rede

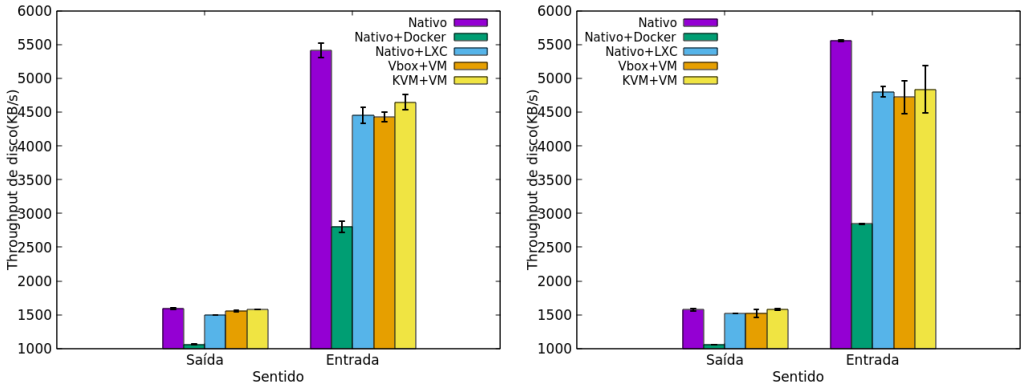
Uma função virtual de rede, como por exemplo um roteador, tem que ser capaz de receber corretamente pacotes e encaminhá-los para a interface de saída. Diante disso, a camada de virtualização deve possuir a menor sobrecarga possível para não afetar o desempenho de rede. A fim de avaliar a sobrecarga de virtualização no desempenho de rede, foi adotada a ferramenta Iperf [McMahon 2003] para medir a vazão com fluxos partindo da tecnologia sob teste com destino para o *host*, como pode ser visto na Figura 2.

A Tabela 2 mostra que o pior desempenho para o protocolos de transporte UDP foi o do VirtualBox. O desempenho do KVM para UDP ficou bem próximo tanto do ambiente nativo quanto das ferramentas que utilizam virtualização leve, que mais uma vez tiveram um bom desempenho.

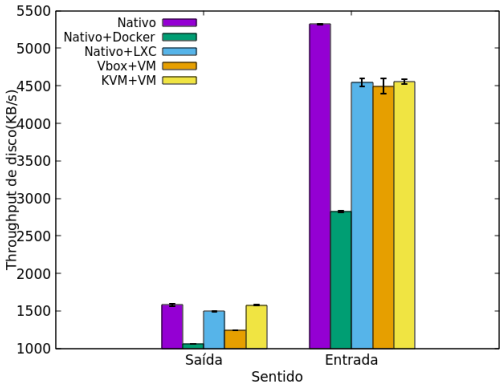
Apesar do Docker e LXC utilizarem uma *bridge* com o uso de *Network Address Translation* (NAT), compartilham o mesmo *kernel* do sistema hospedeiro e acessam os recursos computacionais sem a necessidade de percorrer pilhas adicionais e assim ambos alcançaram um desempenho semelhante ao cenário nativo.

O desempenho do Virtualbox foi inferior aos demais devido a sobrecarga inserida pela emulação da placa que foi a Intel PRO/1000, a qual habilita funções como a *checksum* e *segmentation offloading* que contribuíram para a degradação do desempenho. Além disso, há o esforço para utilização do NAT na interface degrada o desempenho do cenário como podemos observar na Tabela 2.

Por outro lado, o KVM obteve um desempenho melhor que o cenário com Virtualbox devido a utilização de paravirtualização para utilização do dispositivo de rede através



(a) 25 arquivos. (b) 50 arquivos.



(c) 100 arquivos.

Figura 6. Desempenho de Disco: Vazão de acesso ao disco para diferentes quantidades de arquivos criados utilizando o Bonnie++.

da interface *virtio-net* que apesar de diminuir as trocas de contexto entre hospedeiro e visitante, ainda possui sobrecarga elevada em comparação tanto ao ambiente nativo e ao cenário com virtualização leve.

Tabela 2. Desempenho de rede: Vazão total utilizando o Iperf.

Plataforma	Fluxo UDP (Mbps)
Nativo	$815 \pm 0,15$
Nativo+Docker	$814 \pm 0,08$
Nativo+LXC	814 ± 0.2
VirtualBox+VM	$511 \pm 31,59$
KVM+VM	$672 \pm 33,63$

5.5. Desempenho de Aplicação

A aplicação escolhida para análise foi o banco de dados MySQL, uma vez que esta é uma aplicação muito utilizada em diversas verticais tecnológicas. Para avaliar o desempenho das plataformas foi criado em cada uma delas um banco de dados com uma tabela contendo 1 milhão de registros e o benchmarking consiste em realizar consultas de valores com identificadores contidos nessa tabela.

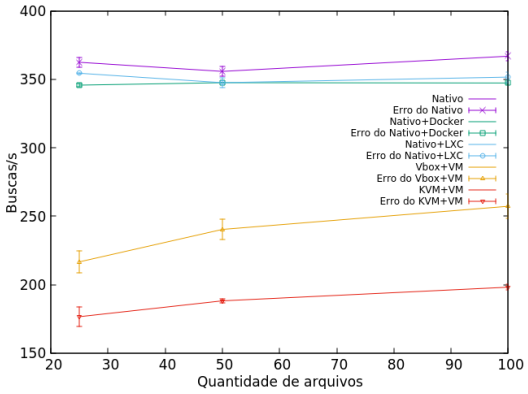


Figura 7. Desempenho de Disco: Taxa de buscas aleatórias em disco feitas com sucesso utilizando o Bonnie++ e 4 processos em paralelo.

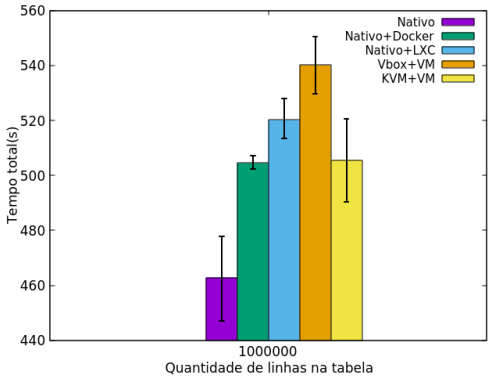


Figura 8. Desempenho de aplicação: Tempo de execução para consultas em uma tabela MySQL com 1 milhão de linhas utilizando o Sysbench.

A métrica considerada para avaliação é o tempo total de execução das operações citadas. Conforme se observa na Figura 8, o desempenho do VirtualBox foi pior, evidenciando uma sobrecarga criada pela virtualização tradicional na execução de aplicações. Esse fraco desempenho poderia degradar uma aplicação em produção, o que torna a virtualização baseada em software uma escolha ruim para aplicações que exijam um alto desempenho de I/O.

O desempenho ruim tanto cenário com LXC quanto o do cenário com Virtualbox evidenciam os gargalos inseridos pelas diferentes técnicas de virtualização no que diz respeito a tempo de acesso a disco, interação com *drivers*, velocidade do CPU e de acesso a memória em ambientes que vão além de benchmarkings sintéticos, ou seja, com cargas reais.

6. Conclusão e Trabalhos Futuros

Soluções baseadas em contêineres desafiam os tradicionais hipervisores, baseados em máquinas virtuais, na computação na nuvem. Essas novas tecnologias são mais leves, facilitando um desenvolvimento mais denso de serviços. Nesse trabalho, foi apresentada uma avaliação de desempenho para diferentes tecnologias de virtualização no sistema operacional Linux.

Além do nível de sobrecarga baixo, do elevado grau de versatilidade e da facilidade de gerenciamento das ferramentas de virtualização serem pontos a favor, ficou evidente através do benchmarking das ferramentas que o desempenho das ferramentas de virtualização leve se aproxima da plataforma nativa na maioria dos casos. Este aspecto é importante para um ambiente de *edge computing*, no qual a arquitetura de microsserviços tende a ganhar cada vez mais força, exigindo um alto desempenho e flexibilidade das plataformas subjacentes.

O objetivo da avaliação foi alcançado ao proporcionar um direcionamento em uma possível implantação de plataformas virtuais, na hora de planejar qual tecnologia de virtualização será utilizada, tendo em vista a sensibilidade da aplicação no que diz

respeito a desempenho. Esse compromisso entre desempenho, flexibilidade e custo tem que ser analisado com mais profundidade em um trabalho futuro.

Referências

Al Jabry, H., Liu, L., Zhu, Y., e Panneerselvam, J. (2014). A critical evaluation of the performance of virtualization technologies. In *Communications and Networking in China (CHINACOM), 2014 9th International Conference on*, Páginas 606–611. IEEE.

Coker, R. (2001). Bonnie++. `http://www.coker.com.au/bonnie++/`. Acessado em 15/02/2017.

Docker, I. (2013). Docker - Build, Ship, and Run Any App, Anywhere. `https://www.docker.com`. Acessado em 01/02/2017.

Eiras, R. S., Couto, R. S., e Rubinstein, M. G. (2016). Performance evaluation of a virtualized HTTP proxy in KVM and Docker. In *Network of the Future (NOF), 2016 7th International Conference on the*, Páginas 1–5. IEEE.

Ismail, B. I., Goortani, E. M., Ab Karim, M. B., Tat, W. M., Setapa, S., Luke, J. Y., e Hoe, O. H. (2015). Evaluation of docker as edge computing platform. In *Open Systems (ICOS), 2015 IEEE Confernece on*, Páginas 130–135. IEEE.

Kopytov, A. (2004). Sysbench. `https://github.com/akopytov/sysbench/`. Acessado em 15/02/2017.

Mattos, D. M., Ferraz, L. H. G., Costa, L., e Duarte, O. C. M. (2012a). Virtual network performance evaluation for future internet architectures. *Journal of Emerging Technologies in Web Intelligence*, 4(4):304–314.

Mattos, D. M., Ferraz, L. H. G., Costa, L. H. M., e Duarte, O. C. (2012b). Evaluating virtual router performance for a pluralist future internet. In *Proceedings of the 3rd International Conference on Information and Communication Systems*, Página 4. ACM.

McCalpin, J. D. (1991). STREAM: Sustainable Memory Bandwidth in High Performance Computers. `https://www.cs.virginia.edu/stream//`. Acessado em 15/02/2017.

McMahon, R. (2003). Iperf2. `https://sourceforge.net/projects/iperf2/`. Acessado em 15/02/2017.

Niu, Z., Xu, H., Tian, Y., Liu, L., Wang, P., e Li, Z. (2016). Benchmarking NFV Software Dataplanes. *arXiv preprint arXiv:1605.05843*.

Pesic, D., Djordjevic, B., e Timcenko, V. (2016). Competition of virtualized ext4, xfs and btrfs filesystems under type-2 hypervisor. In *Telecommunications Forum, 2016 24th*, Páginas 1–4. IEEE.

Varghese, B., Subba, L. T., Thai, L., e Barker, A. (2016). Container-Based Cloud Virtual Machine Benchmarking. Páginas 192–201. IEEE.

Yee, A. J. (2009). Y-cruncher - A Multi-Threaded Pi-Program. `http://www.numberworld.org/y-cruncher/`. Acessado em 15/02/2017.