Avaliação de desempenho de um controlador SDN implementado como uma VNF

Danyel Mendes[1], Marcelo Santos[2], Stenio Fernandes[3]

[1]Instituto Federal Sertão Pernambucano
Serra Talhada – PE, Brasil

[2]Instituto Federal Sertão Pernambucano
Salgueiro – PE, Brasil

[3]Centro de Informática (CIn)
Universidade Federal de Pernambuco (UFPE)
Recife – PE, Brasil

{danyel.mendes, marcelo.santos}@ifsertao-pe.edu.br sflf@cin.ufpe.br

Abstract. *In the present work, we performed experiments with an SDN controller implemented as a virtual network function in the KVM and XEN open source hypervisors and compared the most critical performance parameters in relation to a native scenario in order to measure the degradation of performance caused by Virtualization. We use Cbench to emulate SDN networks and evaluate the performance and latency of the Floodlight controller. We found that controller virtualization in the KVM environment resulted in degraded processing of flows by 29%, showed an increase in response time of 22% and utilized the processor much less than the Xen environment, which despite having overloaded the CPU in 25%, was able to achieve the best performance, response time and the lowest use of RAM.*

Resumo. *No presente trabalho, realizamos experimentos com um controlador SDN implementado como uma função de rede virtual nos hypervisores de código aberto KVM e XEN e comparamos os parâmetros de desempenho mais críticos em relação a um cenário nativo com o propósito de mensurar a degradação de desempenho causada pela virtualização. Utilizamos Cbench para emular redes SDN e avaliar o desempenho e a latência do controlador Floodlight. Descobrimos que a virtualização do controlador no ambiente KVM resultou na degradação do processamento de fluxos em 29%, apresentou um aumento do tempo de resposta de 22% e utilizou bem menos o processador em relação ao ambiente Xen, que apesar de ter sobrecarregado a CPU em 25%, foi capaz de atingir o melhor desempenho, tempo de resposta e a menor utilização de memória RAM.*

1. Introdução

Funções de rede têm sido implementadas por operadores e provedores de serviços em diversos *middleboxes* dedicados proprietários há décadas. Como resultando, temos o aumento de custos OPEX /CAPEX, dificuldade de implantação e gerenciamento de novas funções e serviços. Do ponto de vista dos protocolos e infraestrutura atual da Internet,

temos ainda o problema conhecido como ossificação da internet, fator que tem dificultado o avanço e o crescimento da rede. Neste contexto, a virtualização de funções de rede (NFV - *Network Function Virtualization*) surge como uma solução para a infraestrutura pouco flexível atual da rede que precisa atender as crescentes demandas dos usuários e de novos serviços (JEON; LEE, 2015).

Atualmente, NFV e Redes definidas por software (SDN) são conceitos complementares e estritamente relacionados e a integração entre esses dois paradigmas é denominada de arquitetura *NFV definida por software* (LI; CHEN, 2015). SDN pode beneficiar NFV fornecendo conectividade de rede programável entre funções virtuais e direcionamento de tráfego otimizado no encadeamento de VNFs. Esta integração de tecnologias possibilita maior domínio e agilidade no direcionamento de tráfego, sendo utilizada principalmente em aplicações de encadeamento de serviços, fornecendo também a capacidade de otimização de funções e recursos de rede. Como exemplo, nesse contexto, um controlador SDN pode ser virtualizado como uma NF (*Network Functions* – funções de rede), permitindo a criação e migração dinâmica destas para outros pontos da rede (LI; CHEN, 2015).

Junto com os benefícios trazidos pela virtualização de funções de rede, surgem preocupações com o desempenho, uma vez que fornecer desempenho equiparável as NF implementadas em soluções dedicadas de hardware é um dos grandes desafios de NFV (MIJUMBI et al., 2015), visto que investigações recentes demonstram a degradação de desempenho imposta pelo uso de camadas de virtualização (EIRAS et al., 2016). Neste contexto, em um cenário cuja integração entre SDN e NFV está sendo cada vez mais intensificada, justifica-se a importância de avaliarmos um controlador SDN implementado como uma *função de rede virtual* – VNF, verificando as implicações da virtualização nos parâmetros de desempenho mais críticos. Questões relacionadas a virtualização e desempenho em NFV são pontos discutidos em grupos da IETF/IRTF, especialmente o grupo NFVRG. Especificamente o draft "*Network Virtualization Research Challenges*" (IETF, 2017) traz uma discussão sobre alguns desses desafios.

Apresentamos neste artigo, uma avaliação de desempenho de um controlador SDN implementado como uma função de rede virtualizada. Utilizamos a ferramenta Cbench (LAISSAOUI et al., 2015) como gerador de fluxos e comparamos os resultados obtidos com a execução do controlador diretamente sobre o hardware (sem uma camada adicional de virtualização). O objetivo é mensurar a degradação de desempenho causada pela virtualização nos hypervisores de código aberto KVM (KIVITY et al., 2007) e Xen (BARHAM et al., 2003). Os testes se concentram em medir o desempenho de rede, utilização de CPU total e memória do controlador SDN Floodlight (FLOODLIGHT, 2016). Mensurar o impacto das camadas de virtualização utilizadas sob o controlador Floodlight é a principal contribuição deste trabalho.

O restante deste artigo está estruturado da seguinte forma: Na **sessão 2** apresentamos o referencial teórico necessário para a compreensão dos principais conceitos relacionados com este trabalho. Na **sessão 3** apresentamos alguns trabalhos relacionados mais relevantes, enquanto a metodologia utilizada nesta avaliação e o ambiente de teste são apresentados na **sessão 4**. Na **sessão 5**, apresentamos e discutimos os resultados obtidos. Alguns direcionamentos para trabalhos futuros serão abordados na **sessão 6** e finalmente, apresentaremos as principais conclusões sobre a investigação realizada neste artigo na **sessão 7**.

2. Referencial teórico

2.1. Virtualização

A virtualização possibilita o compartilhamento e a utilização eficiente de recursos computacionais que eram subutilizados em cenários tradicionais, pautados no paradigma servidor por carga de trabalho. Posteriormente, essas cargas de trabalho – que ocupavam uma pequena fatia dos recursos de um servidor físico, passaram a ser hospedadas em computadores virtuais com o devido isolamento e segurança, em um cenário idêntico a um computador real (MANIK, 2016).

A virtualização possibilita o compartilhamento de cargas de trabalho distintas e isoladas em várias instâncias de máquinas virtuais hospedadas em um único host físico. Neste sentido, os recursos computacionais dos servidores são otimizados, proporcionando a redução de custos com hardware e energia, eficiência, confiabilidade, disponibilidade, flexibilidade e segurança (WANG; NG, 2010).

Em um cenário convencional, podemos entender virtualização como uma tecnologia que possibilita a execução de várias instâncias de sistemas operacionais em um único hardware através de uma camada de abstração de software inserida entre o hardware e os sistemas operacionais virtualizados e suas aplicações. Em um ambiente virtualizado temos dois tipos de sistemas: Hospedeiro e visitante. O hospedeiro – *Host*, é o sistema operacional que é instalado de forma nativa na máquina, imediatamente sobre o hardware. O sistema visitante – *Guest*, é instalado em uma máquina virtual – VM, virtualizada pelo sistema hospedeiro (WANG; NG, 2010) (MANIK, 2016).

A camada de abstração comumente é denominada VMM – Monitor de máquina virtual ou hypervisor, e tem a função de gerenciar os dispositivos físicos do computador, permitindo que várias instâncias de computadores virtuais executem simultaneamente através do compartilhamento de hardware, otimizando a utilização de recursos. Um VMM gerencia os recursos de hardware, possibilita a criação de unidades lógicas denominadas máquinas virtuais – VMs e garante o isolamento de recursos e segurança (SAHOO, 2010) (SEMNANIAN et al., 2010) (SORIGA; BARBULESCU, 2013).

2.2. Modos de virtualização

A maneira como as máquinas virtuais são virtualizadas no hypervisor referem-se aos modos de virtualização. Apresentaremos a seguir os dois modos que se relacionam com este trabalho: Virtualização total (full) e para-virtualização.

Virtualização total. Também denominada emulação de hardware, a virtualização total ou *Full*, não exige a modificação do sistema operacional convidado (Guest). Desta maneira, o sistema *Guest* tem a ilusão de estar sendo executado sobre um hardware físico porque a existência de uma camada de abstração de software é transparente para o sistema convidado. Esta abordagem traz uma degradação de desempenho significativa para o sistema operacional visitante porque o comportamento do hardware físico é simulado pelo hypervisor. Antes da execução de qualquer instrução do sistema operacional convidado, o hypervisor realiza um teste para determinar se a instrução é privilegiada ou não, o que acaba reduzindo consideravelmente o desempenho do sistema devido ao custo adicional de processamento. Na virtualização *full*, o hypervisor provê um ambiente virtual para as VMs semelhante a uma réplica do hardware, é executado acima deste e realiza a gestão de recursos físicos da máquina, tais como memória, processamento, E/S e rede (SAHOO, 2010).

Para-virtualização. A para-virtualização é uma técnica que contorna algumas limitações fundamentais da virtualização total, principalmente o desempenho. Diferentemente da virtualização total, a para-virtualização (também denominada virtualização assistida por sistema operacional) requer a modificação do kernel do sistema convidado para que este possa "enxergar" o hypervisor e trabalhar em conjunto com o mesmo. Sahoo et al. (2010) afirma que um sistema convidado para-virtualizado sabe que está sendo virtualizado (SAHOO, 2010). Essa abordagem consegue oferecer um desempenho muito melhor que a virtualização total nos aspectos de CPU, memória e E/S através do suporte especial ao kernel do Linux, que acaba sendo uma desvantagem nos aspectos de compatibilidade e controle de versões de sistemas convidados (BARHAM et al., 2003).

2.3. Kernel-Based Virtual Machine (KVM)

O Kernel-Based Virtual Machine (KIVITY et al., 2007) é um gerenciador de máquinas virtuais (VMM) de código aberto baseado em plataforma x86, desenvolvido e mantido pela Qumranet, Inc. que implementa a virtualização total através das extensões de virtualização da Intel VT-x e AMD-V (SORIGA; BARBULESCU, 2013). O KVM foi integrado ao kernel Linux em 2007 e o transforma em um hypervisor que aproveita as vantagens do kernel padrão, que incorpora as capacidades de virtualização nativa. O KVM consiste em um módulo carregável denominado *kvm.ko* que implementa as funcionalidades fundamentais de virtualização como agendamento do processador e gerenciamento de memória e um módulo específico do processador, *kvm-intel.ko* ou *kvm-amd.ko*. Para simular componentes de hardware de um computador, o KVM utiliza um emulador QEMU (BELLARD, 2005) estreitamente modificado que executa no espaço de utilizador e fornece um modelo de operações de E/S (KIVITY et al., 2007).

2.4. Xen

Xen (BARHAM et al., 2003) é um hypervisor de código aberto muito popular que implementa a para-virtualização de máquinas, é baseado principalmente na plataforma x86 e suporta vários sistemas operacionais, incluindo Linux, Solares, Windows e algumas versões BSD. Xen apresenta um desempenho muito melhor do que a virtualização total quando a máquina hospedeira não possui suporte a virtualização assistida por hardware, mas é preciso que os sistemas convidados sejam alterados para dar suporte à para-virtualização. Xen suporta para-virtualização desde a sua concepção e alcança um desempenho muito próximo ao do host com o custo da modificação do kernel do sistema operacional hospedeiro (SORIGA; BARBULESCU, 2013). Com o surgimento das tecnologias de suporte a virtualização de hardware, Xen também passou a suportar a virtualização total, não necessitando, portanto, da modificação dos sistemas operacionais visitantes (BARHAM et al., 2003).

2.5. Virtualização de funções de rede (NFV)

Através da virtualização de funções de rede (NFV) podemos dissociar as funções de rede de um hardware específico, de forma que serviços podem executar como componentes de software em uma arquitetura virtualizada, eliminando tanto a dependência de hardware específico de determinado fornecedor quanto o fim dos ciclos de vida reduzidos do hardware proporcionados pelos avanços e inovação tecnológica. NFV é um conceito que transfere as funções de rede dos hardwares dedicados de fornecedores específicos para a execução em aplicações baseadas em software. As

funções de rede virtualizadas (VNFs) executam em equipamentos através da virtualização em plataformas consolidadas no mercado, como servidores de alto desempenho. Entre os benefícios principais de NFV estão a melhor eficiência nos gastos de capital (CAPEX) em relação a aquisição de um hardware dedicado, rápida implementação de novos serviços, flexibilidade na definição de funções de rede, redução de energia elétrica, padronização de interfaces e melhoria de eficiência nos processos operacionais (MIJUMBI et al., 2015).

2.6. Redes definidas por software (SDN)

As redes definidas por software é um paradigma que quebra a integração vertical, com a separação radical dos planos de encaminhamento de pacotes e de controle, fornecendo aplicativos com uma visão centralizada abstrata do estado distribuído da rede. Os planos de dados e controle são interligados por interfaces públicas, e possibilitam a programação direta do plano de controle. Desta forma, todas as políticas de rede podem ser implementadas e gerenciadas de forma logicamente centralizada em um controlador, possibilitando a gerencia da rede como uma entidade única e programável. O padrão amplamente utilizado e consolidado para redes SDN é o *OpenFlow* (NUNES et al., 2014). Nas redes SDN, toda a inteligência se encontra logicamente centralizada em controladores baseados em software (plano de controle) e os dispositivos de rede tornam-se elementos de encaminhamento simples de pacotes (plano de dados), que podem ser programados através de uma interface aberta como o OpenFlow (NUNES et al., 2014).

3. Trabalhos relacionados

Desde o surgimento das redes SDN, muitas comparações entre controladores SDN foram realizadas, abordando principalmente análises comparativas de throughput e latência com a utilização de Cbench como em (LAISSAOUI et al., 2015) e (KHATTAK et al., 2014). Esses trabalhos geralmente comparam vários controladores de código aberto, apresentando uma análise quantitativa de rendimento e tempo de resposta. Os cenários quase sempre compreendem a variação do número de switches e threads com o propósito de avaliar a escalabilidade do controlador como em (TOOTOONCHIAN et al., 2012), (SHALIMOV et al., 2013), (ERICKSON, 2013), (ZHAO et al., 2015) e (SALMAN et al., 2016).

Tootoonchian et al. (2012) foi um dos pioneiros nos estudos comparativos de controladores SDN, confrontando apenas NOX-MT, Beacon e Maestro. Em (SHALIMOV et al., 2013), uma ferramenta de benchmark chamada hcprobe foi utilizada em uma análise comparativa da eficácia dos controladores Beacon, NOX, POX, Floodlight, MuL, Maestro e Ryu. Em (LAISSAOUI et al., 2015), Floodlight superou os controladores populares de código aberto Beacon, Pox, Ryu em termos de throughput e latência. Zhao et al.(2015) realizou uma avaliação de desempenho abrangente de controladores SDN e concluiu que Beacon tem o melhor desempenho, enquanto Ryu é melhor nos aspectos de usabilidade e popularidade (ZHAO et al., 2015). Salman et al. (2016) realizou uma avaliação de desempenho e latência de controladores mais recentes utilizando Cbench e concluiu que MuL, Beacon e Maestro tem o melhor tempo de resposta (SALMAN et al., 2016). Outras investigações tiveram foco em ferramentas de benchmark, como o trabalho desenvolvido por (JARSCHEL et al., 2012), que propôs uma ferramenta de benchmark flexível de controladores OpenFlow capaz de criar switches virtuais customizáveis. Por outro lado, (KHATTAK et al., 2014) ressaltou limitações

relativas a Cbench e propôs alterações para adaptá-lo aos cenários de tráfegos de datacenters reais de natureza mais complexa.

Na perspectiva NFV, ultimamente, a IETF – Benchmarking Methodology Working Group (BMWG) têm discutido metodologias de benchmarking para a infraestrutura NFV (NFVI) e funções de rede virtuais em (HUANG et al., 2015). Kim et al. (2015) define métricas de avaliação comparativa de desempenho e confiabilidade de SDN e NVF enquanto (WANG, 2016) se concentra mais em escalabilidade e desempenho de VNFs. Estudos comparativos com cadeias de serviço NFV foram realizados em (BONAFIGLIA et al., 2015) e (ANDERSON et al., 2016). O primeiro apresenta um estudo comparativo das principais tecnologias de virtualização no aspecto do encadeamento de VNFs enquanto o segundo compara o desempenho de rede de VNFs baseadas em contêineres em comparação com a tecnologia de maquinas virtuais.

Alguns trabalhos avaliaram também o desempenho de uma função de rede virtual específica quando a mesma é virtualizada em hypervisores tradicionais em comparação com a recente tecnologia de containers como em (EIRAS et al., 2016) e (MAURICIO et al., 2016). O primeiro conclui que um proxy HTTP virtualizado em Docker solicita HTTP em um tempo menor que em KVM e que um container Docker pode ser duas vezes mais rápido que uma VM baseada em KVM. O segundo implementou um firewall na plataforma OPNFV e verificou que a VNF forneceu capacidade elástica e escalável, atendendo as demandas de tráfego esperadas.

Apesar de algumas investigações recentes terem abordado o desempenho de controladores SDN e a avaliação de VNFs sobre diversas perspectivas, este documento é o primeiro a investigar os overheads de virtualização sobre um controlador SDN virtualizado nos hypervisores KVM e XEN.

4. Metodologia e ambiente de teste

4.1. Ambiente de testes

Os experimentos foram executados utilizando dois computadores conectados por um enlace de 1 Gbps em um ambiente de rede de topologia ponto-a-ponto conforme mostra a **Figura 1**.

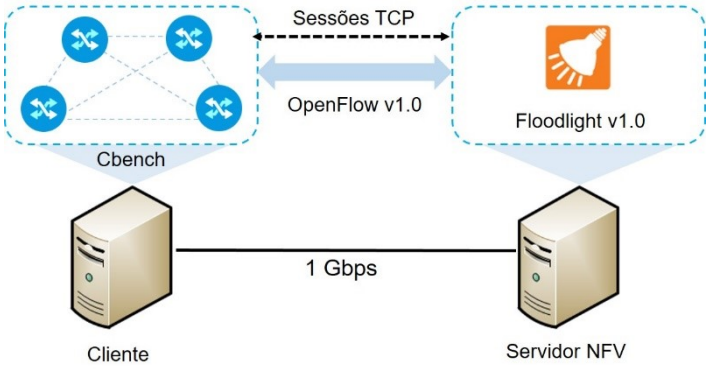

Figura 1. Ambiente de testes utilizado nos experimentos

O servidor possui um único processador Intel Core i5-4590 de 4 núcleos lógicos rodando a 3.30 GHz, 8 GB de RAM e 1 TB de disco rígido enquanto o gerador de tráfego possui 2 processadores Intel (R) Xeon (TM) CPU 3.60 GHz com 4 núcleos lógicos, 2 GB

de RAM e 200 GB de disco rígido. O OpenJDK (IcedTea 2.6.6) foi o ambiente de execução instalado para suportar o controlador SDN no servidor.

O controlador Floodlight (FLOODLIGHT, 2016) versão v1.0 foi configurado na forma padrão de acordo com as diretrizes do site oficial. O sistema operacional Debian GNU/Linux 8 (Jessie) foi instalado no servidor e nas duas máquinas virtuais que hospedaram as VNFs com kernel versão 3.16.0-4-amd64, enquanto o gerador de tráfego hospedou um sistema Debian GNU/Linux 8 com kernel 3.16.0-4-686-pae por se tratar de uma arquitetura i686. As duas máquinas virtuais foram configuradas de modo a serem idênticas ao host que hospedou a função de rede física. A **Tabela 1** exibe a configuração das VNFs virtualizadas nos hypervisores KVM e Xen.

Tabela 1. Configuração das VNFs

VNF	
Sistema	Debian GNU/Linux 8
Linux Kernel	3.16.0-4-amd64
vCPUs	4
vRAM	4 GB
vNICs	1
vDisc	20 GB
Tipo da VNF	Controlador Floodlight v1.0

4.2. Cbench

Cbench (LAISSAOUI et al., 2015) é uma ferramenta de benchmarking de controladores SDN que utiliza o protocolo OpenFlow v1.0. Cbench emula um conjunto configurável de switches que são conectados ao controlador a ser testado de modo que os comutadores são conectados a um número variável de hosts virtuais que podem se comunicar uns com os outros, gerando mensagens do plano de dados para o controlador (SALMAN et al., 2016). A partir da interação entre os hosts virtuais conectados à rede de comutadores, Cbench envia mensagens de solicitação de decisão de encaminhamento (*PACKET_IN*) para o controlador e espera receber a resposta para a solicitação enviada através de mensagens *FLOW_MOD*, sendo capaz de operar em dois modos: *Latency* e *Throughput*. No modo latency, cada switch configurado envia uma única mensagem e espera a resposta para obter o tempo necessário que o controlador utiliza para processar e responder à mensagem, repetindo várias vezes o processo. Desta forma, Cbench calcula a latência média a partir do número total de respostas recebidas no período de teste. No modo *throughput*, Cbench envia o maior número possível de mensagens *PACKET_IN* com o propósito de verificar a capacidade máxima de processamento do controlador (ZHAO et al., 2015).

4.3. Metodologia

Durante a realização dos testes, Cbench foi configurado para emular um número específico de switches capazes de enviar fluxos ao controlador instalado no servidor por um período de tempo determinado. Ao término de cada teste, coletamos os dados gerados pelo Cbench. Simultaneamente a execução de Cbench, os dados de CPU e memória RAM das funções de rede física e virtuais foram registrados pela ferramenta Collectl (COLLECTL, 2016) no servidor NFV.

Neste trabalho, consideramos 3 ambientes de testes: A função de rede física, VNF virtualizada em KVM (*VNF-KVM*) e VNF em máquina virtual XEN (*VNF-XEN*),

conforme apresenta a **Figura 2**. A função de rede física consiste em um controlador SDN instalado diretamente no host, sem a presença de um hypervisor. As funções de rede virtuais foram implantadas em máquinas virtuais (VMs) nos ambientes KVM e XEN, cujas VMs possuem o mesmo sistema operacional e configuração de hardware que o host físico. No ambiente da função física, para fins de comparação com as funções virtuais, foram retirados 4 GB de memória RAM do servidor temporariamente para a execução dos experimentos. Inicialmente, os testes foram feitos no modo *throughput* do Cbench com 1, 2, 4, 8, 16 e 32 switches. Logo após, alteramos o modo para *latency* e os experimentos foram refeitos variando-se o número de comutadores.

Após a conclusão dos testes com a função física, 4 GB de memória foram devolvidos ao host físico e o ambiente da VNF-KVM foi preparado com a criação de uma VM totalmente virtualizada no hypervisor KVM padrão com *QEMU 2.1.2*, de configuração idêntica ao do sistema operacional do host. Em seguida, iniciamos os experimentos com o controlador virtualizado no ambiente KVM, da mesma forma que no ambiente da função física. Após a conclusão dos testes com a VNF-KVM, preparamos o ambiente da VNF-XEN com o hypervisor *Xen 4.4-amd64* na configuração padrão (para-virtualização) e com a mesma configuração de software e hardware da máquina real. Os mesmos testes realizados em VNF-KVM foram feitos na VNF-XEN.

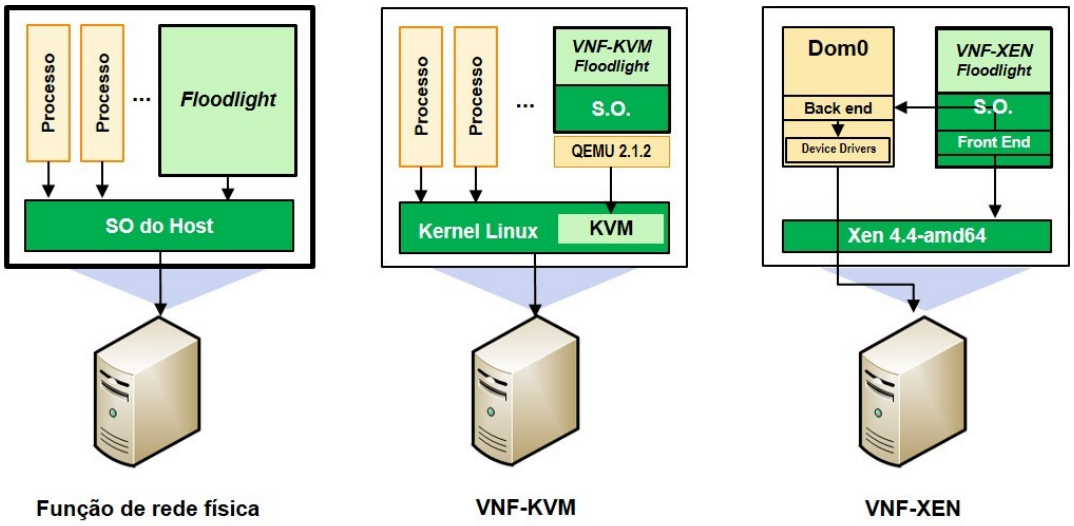

Figura 2. Configurações das funções de rede

A carga de trabalho gerada foi baseada nos experimentos realizados em (ZHAO et al., 2015). Cada experimento durou 8,33 minutos (500 testes de 1 segundo) com um número variável de comutadores, 100 hosts/switch e um modo Cbench (throughput ou latency). Para validar os resultados estatisticamente, repetimos 30 vezes cada experimento e adotamos um intervalo de confiança de 95%.

5. Avaliação de desempenho

Nesta seção, descreveremos os resultados dos experimentos executados com as funções de rede física e virtuais. Inicialmente, apresentaremos os dados de rendimento e tempo de resposta das funções de rede, em seguida vamos expor os resultados da utilização de CPU e por último, apresentaremos os resultados de alocação de memória.

5.1. Desempenho de rede

Enquanto as aplicações de rede geralmente são sensíveis ao processamento, as funções de rede realizam operações intensivas de entrada/saída de rede. Por exemplo, um servidor web que recebe uma solicitação HTTP pode demandar uma quantidade enorme de ciclos de CPU no processamento da solicitação antes de enviar a resposta ao cliente, enquanto um roteador opera basicamente com uma grande quantidade de decisões de encaminhamento, utilizando apenas poucos ciclos de CPU. Um controlador SDN realiza operações intensivas de rede, sendo o desempenho, um fator que merece bastante atenção no cenário NFV, principalmente se considerarmos os overheads de virtualização já amplamente investigados.

No campo de controladores SDN, as duas principais métricas de desempenho amplamente consideradas em diversas investigações são throughput e latência. Throughput refere-se ao rendimento da função de rede, ou seja, a quantidade de fluxos OpenFlow que o controlador é capaz de processar por segundo enquanto a latência mensura o tempo de resposta do controlador a uma mensagem *PACKET_IN* recebida de um comutador.

Tabela 2. Desempenho das funções de rede física e virtuais para 1 switch

Função de rede	T (rps/ms)	L (rps/ms)	T/L	1/L (ms)
Função Física	123,4	2,840	43,45	0,352
VNF-KVM	64,9	2,328	27,87	0,430
VNF-XEN	62,1	3,092	20,08	0,323

A **Tabela 2** mostra os valores de referência (apenas um switch) obtidos para throughput e latência, ambos em milissegundos para todas as funções de rede. O termo "rps/ms" significa respostas por milissegundo e a latência real é 1/L, devido ao fato de Cbench utilizar o número de respostas sequenciais para calcular o atraso. Para Zhao et al. (2015), a proporção T/L corresponde a capacidade do controlador de processar múltiplos fluxos simultâneos e depende de aspectos de design dos controladores SDN (ZHAO et al., 2015). O Floodlight obteve um valor T/L menor nos cenários com virtualização, o que nos permite concluir que a virtualização também é um fator que reduz a capacidade de processamento de fluxos dos controladores. A seguir analisaremos o desempenho de rede do controlador.

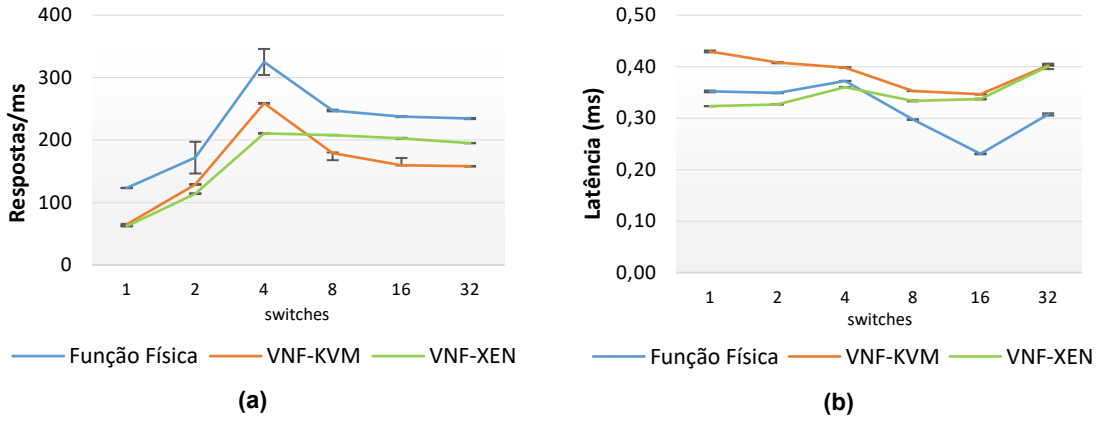

Figura 3. Throughput do controlador (a) e latência (b) para um número variáveis de switches

Throughput. Observando a **Figura 3 (a)** é possível observar que o throughput das funções de rede aumenta consideravelmente até atingir o rendimento máximo com quatro comutadores. A partir desse ponto, o rendimento tende a se manter constante a partir de 16 comutadores (em VNF-XEN esse declínio é mais suave), o que nos permite concluir que o desempenho degrada com o aumento do número de comutadores porque o controlador precisa lidar com um número crescente de solicitações simultâneas do plano de dados. Floodlight cria e mantém uma conexão TCP para cada comutador OpenFlow e estes podem enviar e receber pacotes enquanto houver espaço nos buffers do controlador. Não havendo espaço nos buffers, os pacotes são rejeitados, limitando o desempenho do controlador, que só é capaz de atender a um número de fluxos máximo por segundo. Esses fatores juntos reduzem a capacidade de resposta do controlador e tendem a degradar o seu desempenho com o crescimento da rede. A função de rede física mantém o desempenho bem superior as funções virtuais, independentemente do número de comutadores considerados no experimento.

Latência. É importante ressaltar que a latência apresentada nos gráficos é a latência real por comutador. Assim seu valor é determinado por 1/L, onde L é o número de respostas por segundo. Conforme apresenta a **Figura 3 (b)**, A curva de latência da função de rede tem um comportamento diferente do esperado: A latência é praticamente constante até os 4 switches, decresce em 8 e 16 comutadores e depois aumenta quando a configuração tem 32. Esse comportamento da curva de latência apresentado é semelhante ao do controlador floodlight nos resultados dos experimentos apresentados por (SALMAN et al., 2016). Esse comportamento inesperado de Cbench também foi verificado em (KHATTAK et al., 2014) nos testes de latência realizados com o controlador OpenDaylight, cujo tempo de resposta também diminuiu com o número de switches emulados, sendo um comportamento que precisa ser melhor investigado. A **Figura 3 (b)** mostra de forma clara a diferença dos tempos de resposta do controlador para todos as funções de rede, sendo possível visualizar a degradação de desempenho provocada pela virtualização dos ambientes Xen e KVM. A VNF-KVM obteve o maior tempo de resposta porque utiliza virtualização total, cuja emulação de dispositivos de rede realizada por QEMU no espaço do usuário do host é onerosa e degrada o desempenho. A **Tabela 3** mostra o throughput e a latência do controlador normalizados para a função física.

Tabela 3. Desempenho das funções de rede normalizado para a função física

Desempenho de rede	**Função de Rede**		
	Física	*VNF-KVM*	*VNF-XEN*
Throughput	1	0,71	0,74
Latência	1	1,22	1,09

5.2. Desempenho de CPU

Um conjunto de funções de rede são muito sensíveis à taxa de transferência e latência por que em sua essência operam com o envio/recebimento de pacotes. No entanto, computação também é um fator crítico para o desempenho de boa parte das VNFs, sendo, portanto, necessário analisar o percentual de utilização de CPU do sistema, principalmente em ambientes virtuais que podem impor uma degradação de processamento considerável e limitar o desempenho geral da função de rede. Os valores de desempenho de CPU foram coletados pela ferramenta Collectl e referem-se ao percentual total de utilização do processador e seus núcleos, podendo chegar a 100%.

A **Figura 4 (a)** apresenta os resultados de utilização de CPU total para os testes de throughput. Analisando o gráfico, verificamos que a utilização de CPU das funções de rede aumenta gradualmente até 4 comutadores. Para mais de 4 comutadores, a VNF-XEN continua com a CPU sobrecarregada enquanto as demais funções de rede utilizam cerca de 65% da CPU a partir de 8 comutadores. Desta forma, podemos concluir que há uma relação entre o aumento do número de switches emulados e a utilização de CPU até 4 comutadores, mas o crescimento da rede e o consequente aumento de demandas do plano de dados diminuem a utilização do processador, com exceção da VNF-XEN, que aproveita bem a CPU no processamento de fluxos, apesar de sobrecarregar o processador com o ambiente para-virtualizado.

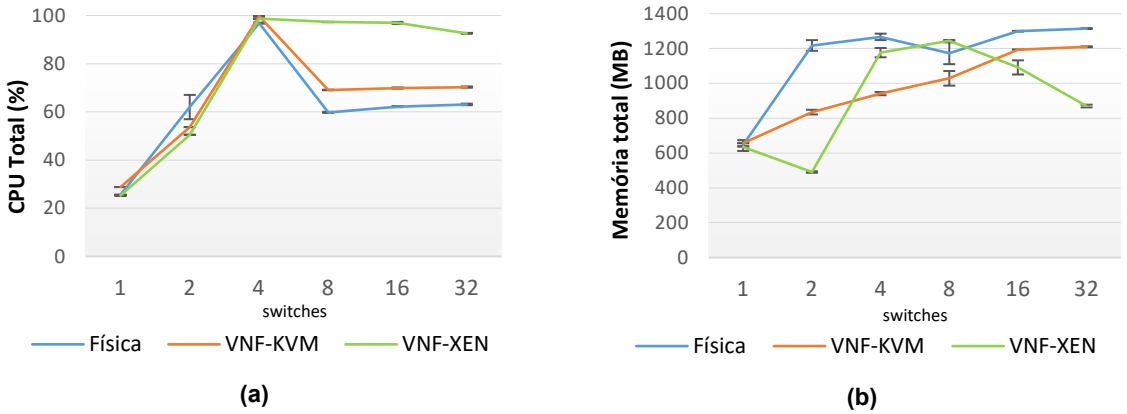

Figura 4. Desempenho de CPU (a) e memória (b) das funções de rede

Conforme podemos observar na **Figura 3 (a)**, a VNF-XEN tem o seu rendimento máximo em 4 comutadores e estabiliza a partir desse ponto devido ao esgotamento dos recursos da CPU para o processamento de fluxos, uma vez que o processador passa a ser bastante utilizado também no ambiente de virtualização, nas interações entre a VNF e o domínio privilegiado Xen. Esse fator limita o desempenho da VNF, que trabalha conjuntamente com o domínio privilegiado Dom0 para obter um melhor desempenho. O agendamento de CPU para a VNF-XEN é controlado por Dom0 ao invés de ser realizado diretamente pelo kernel do sistema como em KVM. Desta forma, a forte interação com Dom0 é o principal fator que degrada o processamento da VNF-XEN.

Por outro lado, o agendamento de processos (VMs) realizado pelo próprio KVM no host com privilégios de modo Kernel, resulta em baixa degradação de CPU na VNF-KVM (6%). O KVM é um módulo que se integra ao Kernel Linux, transformando-o em um hypervisor, e as VMs são processos que aguardam para serem escalonados pelo núcleo. A **Tabela 4** mostra a média de utilização de CPU total das funções de rede.

Tabela 4. Desempenho de CPU das funções de rede

CPU Total	**Função de Rede**		
	Função Física	VNF-KVM	VNF-XEN
CPU – Modo throughput	62%	65%	77%

5.3. Desempenho de Memória

Gerenciar a utilização de memória RAM com eficiência é um fator decisivo para o desempenho de qualquer sistema computacional, principalmente em cenários cujos

recursos são limitados. Aspectos de alocação de memória precisam ser considerados em avaliação de performance de VNFs porque o desempenho geral do sistema pode ser afetado negativamente se o mal gerenciamento da memória RAM resultar em acessos a memória segundaria (que é mais lenta). Essa situação se agrava ainda mais em cenários virtualizados porque a E/S de disco ainda é um grande gargalo de desempenho de sistemas virtualizados (CHE et al., 2010).

Conforme mostra a **Figura 4 (b)**, de maneira geral, a alocação de memória tem um comportamento esperado: a utilização aumenta com o número de switches porque o controlador precisa processar um número maior de solicitações de plano de dados enviado pelos comutadores. No entanto, a alocação de memória tende a ficar estável quando a rede tem mais de 16 comutadores, exceto para VNF-XEN que não seguiu um padrão de alocação de memória RAM.

A única métrica não afetada negativamente pela virtualização da função de rede foi a memória, uma vez que as VNFs tiveram o desempenho degradado pela virtualização, processaram menos fluxos e consequentemente alocaram menos memória. A VNF-XEN obteve menor utilização de memória em relação a VNF-KVM provavelmente devido a otimização do acesso a memória realizado por Xen, pois a VNF não precisa notificar o hypervisor sobre atualizações nas tabelas de páginas a cada acesso a memória virtual. A **Tabela 5** mostra a alocação de memória média das funções de rede.

Tabela 5. Desempenho de memória das funções de rede

Memória Total	**Função de Rede**		
	Física	*VNF-KVM*	*VNF-XEN*
Memória – Modo throughput	1153 MB ≈ **28%**	978 MB ≈ **24%**	918 MB ≈ **22%**

6. Conclusões

No presente trabalho, realizamos experimentos com um controlador SDN implementado como funções virtuais nos ambientes KVM e XEN, e comparamos os parâmetros de desempenho mais críticos em relação a um cenário nativo com o propósito de mensurar a degradação de desempenho causada pela virtualização de funções de rede.

Conforme mostra a **Tabela 3**, a virtualização do controlador SDN resulta em perdas significativas de desempenho, uma vez que as VNFs implementadas em KVM e Xen atingiram apenas 71% e 74% do rendimento da função física. O tempo de resposta do controlador também foi bastante prejudicado pela virtualização, sendo 22% maior no ambiente KVM e 9% em Xen. O controlador virtualizado em Xen obteve melhor rendimento e tempo de resposta porque utiliza para-virtualização enquanto KVM emula dispositivos de E/S no espaço de usuário host com QEMU.

Referente ao desempenho de CPU, a VNF apresentou uma degradação de 25% no ambiente Xen e 6% em KVM em relação a função física, o que nos permite concluir que a VNF-XEN sobrecarregou a CPU porque processou uma maior quantidade de fluxos e utilizou mais tempo de CPU no ambiente virtualizado. Acreditamos que a forte interação inter domínio e o agendamento de máquinas virtuais de Xen contribuíram com a sobrecarrega de CPU na VNF. Por outro lado, KVM – que é um hypervisor integrado ao kernel Linux, insere uma pequena sobrecarga de processamento na VNF (6%) por que o agendamento de CPU para as VMs é realizado pelo próprio kernel, sugerindo a emulação de E/S como o grande responsável pela degradação de desempenho da VNF-KVM.

Os resultados dos experimentos mostram que a degradação de desempenho do controlador resultou em uma menor utilização de memória RAM nas VNFs. A VNF

alocou 24% da memória total em KVM e 22% em Xen representando, portanto, uma utilização de 6% e 4% menor em relação a função física. As otimizações de acesso a memória realizadas pelo hypervisor Xen podem ajudar a explicar a menor alocação de RAM na respectiva VNF.

Apesar da significativa sobrecarga de CPU, concluímos que, quando comparado a KVM, o hypervisor Xen é a melhor opção para a virtualização de controladores SDN. Nossos resultados mostraram que, quando virtualizado no ambiente Xen, um controlador SDN tem a menor degradação de desempenho, o melhor tempo de resposta e a menor alocação de memória RAM. No entanto, mesmo com a paravirtualização do ambiente Xen, a virtualização de um controlador SDN implica em uma considerável degradação de desempenho, tornando-se necessário avaliar tecnologias de virtualização mais "leves" e eficientes.

7. Trabalhos futuros

Futuramente, pretendemos avaliar o desempenho dos controladores baseado em Kernel Linux (IVASHCHENKO et al., 2014) e Beacon (ERICKSON, 2013), implementados como VNFs nos hypervisores Xen (virtualização total assistida por Hardware – HVM), KVM com drivers paravirtualizados *Virtio* (RUSSELL, 2008) e contêineres (FELTER et al., 2015), utilizando um maior número de comutadores OpenFlow. Pretendemos também, demonstrar que *Unikernels* (XAVIER et al., 2016) podem ser utilizados com tecnologias alternativas de implementação de dispositivos de rede como Macvlan (HICUBE, 2017) e SR-IOV (ANDERSON et al., 2016) para melhorar o desempenho de controladores SDN virtualizados.

Referencias

ANDERSON, J.; AGARWAL, U.; LI, H.; SOFTWARE, D. Performance Considerations of Network Functions Virtualization using Containers. , p. 1–25, 2016.

BARHAM, P.; DRAGOVIC, B.; FRASER, K.; et al. Xen and the art of virtualization. **ACM SIGOPS Operating Systems Review**, v. 37, n. 5, p. 164, 2003.

BELLARD, F. QEMU , a Fast and Portable Dynamic Translator. **USENIX Annual Technical Conference. Proceedings of the 2005 Conference on**, p. 41–46, 2005.

BONAFIGLIA, R.; CERRATO, I.; CIACCIA, F.; NEMIROVSKY, M.; RISSO, F. Assessing the Performance of Virtualization Technologies for NFV: a Preliminary Benchmarking. , p. 67–72, 2015.

CHE, J.; YU, Y.; SHI, C.; LIN, W. A synthetical performance evaluation of OpenVZ, Xen and KVM. **Proceedings - 2010 IEEE Asia-Pacific Services Computing Conference, APSCC 2010**, p. 587–594, 2010.

COLLECTL. Collectl. Disponível em: <http://collectl.sourceforge.net/>. Acesso em: 2/2/2017.

EIRAS, R. S. V; COUTO, R. S.; RUBINSTEIN, M. G. Performance evaluation of a virtualized HTTP proxy in KVM and Docker. 2016 7th International Conference on the Network of the Future (NOF). **Anais...** . p.1–5, 2016. IEEE.

ERICKSON, D. The beacon openflow controller. **Proceedings of the second ACM SIGCOMM workshop …**, p. 13–18, 2013. Disponível em:

<http://dl.acm.org/citation.cfm?id=2491189>. .

FELTER, W.; FERREIRA, A.; RAJAMONY, R.; RUBIO, J. An updated performance comparison of virtual machines and Linux containers. **2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)**, p. 171–172, 2015.

FLOODLIGHT, P. Project Floodlight: Open Source Software for Building Software-Defined Networks. Disponível em: <http://www.projectfloodlight.org/>. Acesso em: 25/2/2017.

HICUBE. Bridge vs Macvlan. Disponível em: <http://hicu.be/bridge-vs-macvlan>. Acesso em: 15/3/2017.

HUANG, L.; GU, R.; LIU, D.; et al. Benchmarking Methodology for Virtualization Network Performance. Disponível em: <https://tools.ietf.org/html/draft-huang-bmwg-virtual-network-performance-00>. Acesso em: 26/3/2017.

IETF. Network Virtualization Research Challenges. Disponível em: <https://tools.ietf.org/html/draft-irtf-nfvrg-gaps-network-virtualization-02#page-15>. Acesso em: 24/3/2017.

IVASHCHENKO, P.; SHALIMOV, A.; SMELIANSKY, R. High performance in-kernel SDN/OpenFlow controller. **Usenix.Org**, p. 1–2, 2014.

JARSCHEL, M.; LEHRIEDER, F.; MAGYARI, Z.; PRIES, R. A flexible OpenFlow-controller benchmark. **Proceedings - European Workshop on Software Defined Networks, EWSDN 2012**, p. 48–53, 2012.

JEON, H.; LEE, B. Network Service Chaining Challenges for VNF Outsourcing in Network Function Virtualization. , p. 819–821, 2015.

KHATTAK, Z. K.; AWAIS, M.; IQBAL, A. Performance evaluation of OpenDaylight SDN controller. **Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS**, v. 2015–April, p. 671–676, 2014.

KIM, T.; KOO, T.; PAIK, E. SDN and NFV benchmarking for performance and reliability. **2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)**, p. 600–603, 2015.

KIVITY, A.; LUBLIN, U.; LIGUORI, A.; KAMAY, Y.; LAOR, D. kvm: the Linux virtual machine monitor. **Proceedings of the Linux Symposium**, v. 1, p. 225–230, 2007.

LAISSAOUI, C.; IDBOUFKER, N.; ELASSALI, R.; BAAMRANI, K. EL. A measurement of the response times of various OpenFlow / SDN controllers with CBench. , p. 0–1, 2015.

LI, Y.; CHEN, M. Software-Defined Network Function Virtualization: A Survey. **IEEE Access**, v. 3, p. 2542–2553, 2015.

MANIK, V. K. Performance Comparison of Commercial VMM : , p. 1771–1775, 2016.

MAURICIO, L. A. F.; RUBINSTEIN, M. G.; DUARTE, O. C. M. B. Proposing and evaluating the performance of a firewall implemented as a virtualized network function. 2016 7th International Conference on the Network of the Future (NOF). **Anais...** . p.1–3, 2016. IEEE.

MIJUMBI, R.; SERRAT, J.; GORRICHO, J.-L.; et al. Network Function Virtualization: State-of-the-art and Research Challenges. **IEEE Communications Surveys & Tutorials**, , n. c, p. 1–1, 2015.

NUNES, B. A. A.; MENDONCA, M.; NGUYEN, X. N.; OBRACZKA, K.; TURLETTI, T. A survey of software-defined networking: Past, present, and future of programmable networks. **IEEE Communications Surveys and Tutorials**, v. 16, n. 3, p. 1617–1634, 2014.

RUSSELL, R. virtio: Towards a De-Facto Standard For Virtual I / O Devices. **ACM SIGOPS Operating Systems Review**, v. 42, p. 95–103, 2008.

SAHOO, J. Virtualization : A Survey On Concepts , Taxonomy And Associated Security Issues. **IEEE Communications Surveys & Tutorials**, 2010.

SALMAN, O.; ELHAJJ, I. H.; KAYSSI, A.; CHEHAB, A. SDN Controllers : A Comparative Study. , , n. 978, p. 18–20, 2016.

SEMNANIAN, A. A.; PHAM, J.; ENGLERT, B.; WU, X. Virtualization technology and its impact on computer hardware architecture. **Proceedings - 2011 8th International Conference on Information Technology: New Generations, ITNG 2011**, p. 719–724, 2010.

SHALIMOV, A.; ZIMARINA, D.; PASHKOV, V. Advanced Study of SDN / OpenFlow controllers. **Proceeding CEE-SECR '13 Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia**, 2013.

SORIGA, S. G.; BARBULESCU, M. A comparison of the performance and scalability of Xen and KVM hypervisors. **Proceedings - RoEduNet IEEE International Conference**, 2013.

TOOTOONCHIAN, A.; GORBUNOV, S.; GANJALI, Y.; CASADO, M.; SHERWOOD, R. On controller performance in software-defined networks. **Proceeding Hot-ICE'12 Proceedings of the 2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services**, p. 10–10, 2012.

WANG, C. Toward High-Performance and Scalable Network Functions Virtualization. , 2016.

WANG, G.; NG, T. S. E. The impact of virtualization on network performance of Amazon EC2 Data Center. **Proceedings - IEEE INFOCOM**, 2010.

XAVIER, B.; FERRETO, T.; JERSAK, L. Time Provisioning Evaluation of KVM, Docker and Unikernels in a Cloud Platform. **Proceedings - 2016 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2016**, p. 277–280, 2016.

ZHAO, Y.; IANNONE, L.; RIGUIDEL, M. On the Performance of SDN Controllers : A Reality Check. , p. 79–85, 2015.