Avaliação experimental das plataformas Xen, KVM e OpenFlow no roteamento de pacotes

Leopoldo A. F. Mauricio e Marcelo G. Rubinstein

***Resumo*—O objetivo deste trabalho é avaliar qualitativamente o desempenho de ambientes virtuais de roteamento construídos com as plataformas de virtualização KVM e Xen, combinadas com o protocolo OpenFlow que permite criar redes definidas por softwares (SDNs). Inicialmente o Xen e o KVM foram instalados em servidores x86 modernos para que seus desempenhos de CPU, acesso à memória RAM e operações de entrada/saída de dados (roteamento de pacotes) fossem comparados. Os resultados obtidos mostram que, apesar de ser uma plataforma de virtualização completa, o KVM com Virtio possui melhor desempenho de memória, de CPU e no roteamento de pacotes que o Xen. Além disso, o desempenho da VM Xen é melhor que o obtido no Xen roteado, tornando-se praticamente semelhante ao do KVM e do Linux nativo, quando combinado com o OpenFlow para a operar com separação de planos.**

Palavras-Chave— Roteamento, Desempenho, OpenFlow, Xen, KVM.

***Abstract*— This paper aims at qualitatively evaluating the performance of routing virtual environments built with KVM and Xen virtualization platforms, combined with the OpenFlow protocol, which is used for creating Software-Defined Networking (SDN). Initially, Xen and KVM were installed on modern x86 servers to compare their CPU, RAM memory access and networking performances. Results show that KVM, when Virtio is used, despite being a full virtualization platform, has better memory, CPU and networking performance than Xen. Furthermore, the Xen VM packet routing performance is better than Xen routed, and almost equal to the KVM and native Linux, when they are installed with OpenFlow to separate the router control plan from router data plan.**

Keywords—Routing, performance, OpenFlow, Xen, KVM.

I. INTRODUÇÃO

Atualmente, existe um interesse na virtualização de servidores para otimizar o espaço físico utilizado em racks, diminuir o consumo e os gastos com energia elétrica e refrigeração em Centros de Processamento de Dados (CPDs) e Datacenters, etc. No entanto, de acordo com os pluralistas, a virtualização também nos permite resolver problemas antigos que o núcleo da rede possui, tais como os problemas de segurança, privacidade, suporte a nós móveis, etc. [14]. Com a virtualização, é possível partir do zero na definição de novas lógicas de encaminhamento de pacotes e novos protocolos para o núcleo da rede [11], que podem ser mais adequados às necessidades das aplicações atuais, sem que seja necessário interromper ou afetar a lógica atualmente implementada pelo protocolo IP no núcleo da Internet [5].

De acordo com o Gartner, a virtualização na plataforma x86 é responsável por mais de 50% da carga útil dos servidores em operação na Internet [4]. Além disso, é possível criar diversas redes virtuais em máquinas x86 com grande capacidade de hardware. Portanto, é importante identificar a melhor plataforma de virtualização para a criação de redes virtuais com bom desempenho. Por isso, nesse trabalho, através de medições, comparamos os desempenhos de memória, CPU e de rede das plataformas de virtualização de hardware Xen e KVM, instaladas em máquinas x86. Em seguida, o OpenFlow foi instalado junto com o Xen e com o KVM, separando os planos de controle e de dados. Dessa forma, foram obtidos melhores desempenhos no roteamento de pacotes quando comparados ao Xen roteado e ao KVM com Virtio, que permite que os drivers de rede da VM sejam executados no modo de kernel do Sistema Operacional (SO).

II. CARACTERÍSTICAS DO KVM, DO XEN E DO OPENFLOW

Um virtualizador de sistemas (hipervisor) pode operar de diferentes formas para permitir o funcionamento de múltiplas instâncias de dispositivos virtuais. Na virtualização completa, o hipervisor simula (ou emula) o hardware da máquina para os SOs das VMs (ou SOs visitantes) agindo como uma camada de controle entre os dispositivos físicos e o SO virtualizado, para impedir que os SOs das VMs solicitem acesso direto aos dispositivos físicos [3] [8]. Na para-virtualização, as VMs conhecem todos os endereços de hardware e os drivers de dispositivos do SO virtualizado conseguem interagir com o hipervisor para que ele permita o acesso direto, porém controlado, dos SOs convidados aos dispositivos físicos [2]. Portanto, espera-se que a sobrecarga do hipervisor que realiza a para-virtualização seja menor que a do hipervisor que precisa emular os dispositivos físicos (virtualização completa), pois o hipervisor que para-virtualiza um hardware executa uma tarefa mais simples. Por isso, espera-se também que o desempenho das VMs de uma solução desse modo de operação seja melhor que o encontrado na virtualização completa [9].

Para fazer o núcleo do Linux nativo agir como um hipervisor, que no KVM fica localizado acima da camada física (Fig. 1), o código do KVM é integrado ao código do kernel [6]. Por isso, quando o kernel do Linux é carregado, o KVM também é executado e ele depende sempre de um processo QEMU, que é um virtualizador e um emulador de hardware, em execução dentro do SO virtual (Fig. 1) para emular o espaço de usuário (modos de acesso da plataforma x86) [9], a fim de que a virtualização dos dispositivos de E/S seja realizada. O KVM é uma solução de virtualização completa, uma vez que as chamadas de sistema e todas as operações de E/S demandadas pelas VMs são gerenciadas pelo QEMU, que as direciona corretamente para o hardware. A árvore "/dev/kvm" mostrada na Fig. 1, com espaços de endereços exclusivos para cada SO convidado, é exportada pelo hipervisor KVM que, por ser executado dentro do núcleo do Linux, pode apresentar desempenho melhor que o encontrado em uma solução de virtualização completa padrão. Isso pode ser feito habilitando-se o Virtio da biblioteca de programação libvirt, para que os drivers do dispositivo de hardware que o SO da VM deseja acessar sejam informados que estão sendo executados em um ambiente virtual. Dessa forma, os drivers

Leopoldo Mauricio, Grupo de Teleinformática e Automação - PEE/COPPE - Universidade Federal do Rio de Janeiro e Globo.com. Marcelo Rubinstein, PEL/UERJ - Universidade do Estado do Rio de Janeiro. E-mails: Leopoldo{@gta.ufrj.br, @corp.globo.com}, rubi@uerj.br. Este trabalho foi parcialmente financiado pelo CNPq e pela Faperj.

interagem com o QEMU (Fig. 1), que lhes permite acessar o modo de kernel [9] do hospedeiro, para que suas operações de E/S de rede e disco possam ser realizadas no modo privilegiado [7] [9].

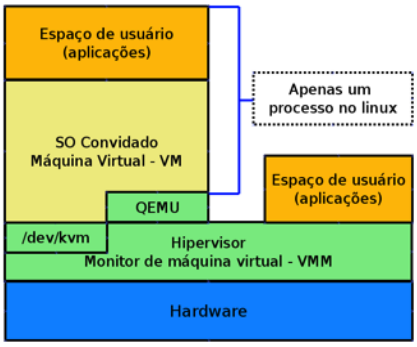

Fig. 1. Arquitetura da plataforma de virtualização KVM.

A arquitetura do Xen (Fig. 2) é diferente da encontrada no KVM. Seu hipervisor fica acima da camada física; sobre ele há um domínio de drivers privilegiado, conhecido como dom0, e diversos domínios convidados (não privilegiados), conhecidos como domUs, podem ser criados. O dom0 do Xen possui acesso total ao hardware, armazena os drivers dos dispositivos físicos, hospeda o software de gerenciamento que possibilita a criação e remoção de VMs e "enxerga" as interfaces de rede da máquina física. Enquanto ele possui acesso total ao hardware, os domUs utilizam drivers virtuais (Fig. 2) para se comunicarem com os recursos físicos através do hipervisor [2].

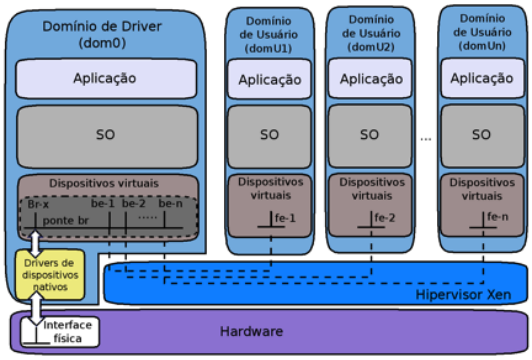

Fig. 2. Arquitetura da plataforma de virtualização Xen.

Ao hipervisor do Xen cabe agendar o uso da CPU entre as VMs, filtrar pacotes de rede antes do encaminhamento, aplicar controle de acesso na leitura de blocos de dados etc. Ele realiza esse trabalho controlando o acesso dos domUs aos recursos físicos da máquina, substituindo as chamadas de sistema dos SOs convidados, que não são encaminhadas diretamente para o hardware, pelas chamadas do Xen, que são conhecidas como *hypercalls* [2]. Por isso, ainda que na para-virtualização um domU conheça os endereços de hardware de seus vizinhos, cada VM é capaz de utilizar apenas a memória e o espaço de disco que lhe foram alocados. Além disso, o Xen pode utilizar pontes (*bridges*) para interligar as interfaces de rede das VMs ou a arquitetura roteada. Egi et al. [2] e Fernandes e Duarte [3] mostraram que o desempenho do Xen roteado é melhor que o do Xen em modo bridge porque o mecanismo de roteamento do kernel é mais eficiente e menos custoso que o implementado na transmissão de pacotes usando pontes. Por este motivo, em todas as avaliações apresentadas neste artigo (no Linux, no Xen e no KVM) não foram utilizadas bridges.

A lógica de repasse de um comutador OpenFlow pode ser completamente diferente da implementada hoje na Internet, onde o roteamento é realizado em função do endereço IP de destino inserido no cabeçalho do pacote, pois ela é função da combinação de um conjunto de tuplas que estão contidas no cabeçalho do fluxo. Além disso, o OpenFlow [5] permite que o plano de dados (tabelas de encaminhamento e roteamento) de um roteador possa ser compartilhado entre múltiplas redes virtuais que possuem lógicas de encaminhamento distintas. O plano de dados fica localizado no comutador, o plano de controle em um nó conhecido como controlador OpenFlow e a comunicação entre o controlador e o comutador, que o controlador gerencia e monitora, é realizada através de um canal seguro de comunicação que interliga esses equipamentos.

Todas as vezes que os primeiros pacotes de um fluxo de dados chegam a um comutador OpenFlow, ele verifica se o controlador já criou uma entrada para esse tráfego em sua tabela de fluxo. Caso isso ainda não tenha sido feito, os primeiros pacotes são encaminhados para o controlador, através do canal seguro de comunicação, para que ele defina por quais comutadores e roteadores o encaminhamento deve ser realizado. Ou seja, o controlador processa os primeiros pacotes para inserir as rotas necessárias para esse fluxo de dados nas tabelas de repasse (no plano de dados) dos comutadores OpenFlow que fazem parte do caminho a ser percorrido [1]. Assim, com o OpenFlow, o plano de dados localizado no comutador, que possui a lógica e as regras de encaminhamento dos pacotes, está separado do plano de controle, que fica no controlador; e apenas o nó controlador possui o plano de controle para que só ele seja capaz de gerenciar as tabelas de repasse dos planos de dados dos comutadores OpenFlow [8].

III. EFICIÊNCIA DO XEN E DO KVM

A eficiência de um roteador é função da capacidade que ele possui para realizar operações de E/S de rede, tanto de encaminhamento quanto de roteamento de pacotes. E em máquinas x86, isso varia em função das capacidades de processamento e memória do hardware, uma vez que as tabelas de roteamento são manipuladas na memória dos roteadores criados, e da taxa de transmissão das interfaces de rede da máquina x86. Além disso, por demandar poucas operações de E/S de disco, o bom desempenho de um roteador virtual criado nesta plataforma depende pouco da capacidade de acesso a disco da solução de virtualização. Portanto, espera-se que quanto menores forem as sobrecargas de memória e CPU e quanto maior for a capacidade de encaminhamento e roteamento de pacotes em uma plataforma de virtualização, mais eficientes serão os roteadores virtuais criados em suas VMs.

Dessa forma, a sobrecarga dos hipervisores do Xen e do KVM sobre a CPU e sobre a memória foi medida, respectivamente, com os benchmarks Super Pi [13] e STREAM 5.1 [12]. Com o Super Pi, diversas operações aritméticas foram realizadas para que o número Pi com 2^{22} casas decimais fosse calculado, a fim de sobrecarregar o uso de CPU. O benchmark STREAM 5.1 foi utilizado, por permitir que a largura de banda sustentável, e não apenas picos de largura de banda ótimos de acesso à memória, seja medida [12]. Para isso, ele exibe as medidas da taxa de transferência de blocos de dados de uma posição da memória para outra, depois que quatro operações de leitura e escrita na RAM são realizadas. Como as VMs foram criadas com quatro CPUs, o código otimizado do STREAM foi compilado com o recurso de programação paralela OpenMP (*Open Multi-Processing*) para que múltiplas threads pudessem ser

executadas ocupando todas as CPUs durante o benchmark de memória [10]. Além disso, os tempos de execução foram medidos com um relógio externo, pois as medições realizadas dentro dos ambientes virtualizados não são confiáveis por se basearem na emulação dos dispositivos de tempo reais, uma vez que os SOs das VMs não conseguem acessar de forma direta as interrupções e os dispositivos do relógio físico [8]. Cada uma das quatro operações de leitura e escrita na RAM foi realizada cem vezes em cada uma das 15 rodadas de teste, com a melhor largura de banda obtida sendo registrada. O primeiro dos cem valores medidos foi excluído, porque com a memória vazia, um pico de largura de banda ótimo não sustentável poderia mascarar o resultado dos testes.

3.1 Capacidade de CPU e Memória

Três servidores x86 Dell 2950 de mesma configuração de hardware foram utilizados nos testes. Cada servidor possuía 1 processador Intel Xeon 5300 de quatro núcleos com relógio de 3 GHz, 4 MB de memória cache por núcleo, 8 GB de memória RAM, 876 GB de disco rígido e duas interfaces de rede de 1 Gbit/s. As três máquinas foram instaladas com o CentOS release 6.4 e kernel Linux 2.6.32-358.2.1 de 64 bits. O KVM desta versão de CentOS e a versão 4.2.1 do Xen para-virtualizado foram utilizados nos testes. Os resultados dos benchmarks de CPU e memória executados em uma VM Xen e em uma VM KVM, ambas com 2 GB de RAM e 4 CPUs virtuais [9], foram comparados com os do Linux nativo de 64 bits. A média de quinze baterias de testes, com um intervalo de confiança de 95%, é apresentada na Fig. 3.

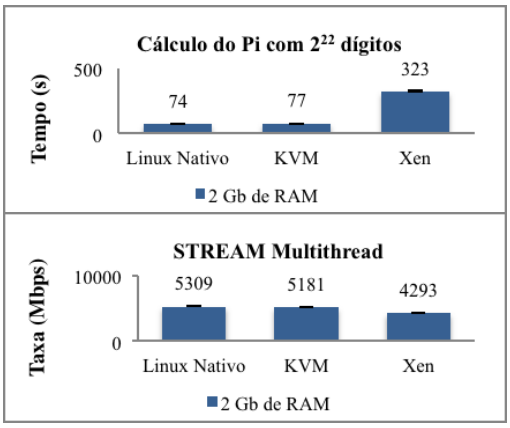

Fig. 3. Desempenho de memória e cálculo do Pi com 2^{22} decimais.

Os resultados mostram que, apesar de utilizar o QEMU para virtualizar completamente o hardware, os desempenhos de CPU e memória do KVM foram os que mais se aproximaram dos encontrados no Linux nativo. Além disso, a sobrecarga, tanto de CPU quanto de memória do hipervisor do KVM foi menor que a do hipervisor do Xen. Isso aconteceu porque o código do KVM é integrado ao código do kernel do Linux, para que o próprio kernel atue como hipervisor quando o KVM é habilitado. No KVM, as VMs são processos aguardando para serem escalonados pelo agendador de tarefas do SO hospedeiro, para utilizar a CPU. A sobrecarga de CPU do hipervisor Xen é maior que a do KVM porque ele não é implementado no kernel do Linux nativo. Por isso, apesar do hipervisor do Xen ser um módulo executado no modo de acesso de núcleo, ele não é capaz de escalonar o uso da CPU para suas VMs de forma direta como acontece no KVM. Ou seja, primeiro ele mesmo precisa ser escalonado pelo agendador de tarefas do SO hospedeiro, para então encaminhar de forma controlada as chamadas de sistema que os SOs das VMs fizeram para a CPU. Os resultados também mostram que o controle implementado pelo hipervisor do Xen na paginação de memória é mais custoso que o realizado pela plataforma de virtualização KVM. No KVM, é o *kernel* do Linux nativo, agindo como hipervisor, que garante a interação de um SO virtualizado com as tabelas de página de memória do *hardware*. E isto é realizado sempre no modo de núcleo da plataforma x86, apesar da emulação do *hardware*. Essa é a característica que faz com que o hipervisor do KVM seja mais eficiente que o do Xen no acesso à memória.

3.2 Capacidade de Encaminhamento e Roteamento

Depois de verificar que o KVM apresenta sobrecarga de memória e CPU menores que a do Xen, executamos o Iperf para comparar a capacidade de encaminhamento e de roteamento de pacotes das VMs Xen e KVM de 2 GB de RAM e 4 CPUs virtuais. Primeiro medimos a taxa de encaminhamento em uma ligação ponto-a-ponto, entre uma máquina geradora de tráfego e uma máquina receptora. A máquina geradora de tráfego no primeiro instante foi o próprio Linux nativo, depois a VM Xen e por fim a VM KVM, e na receptora de tráfego foi instalado o Linux nativo de 64 bits. Foram utilizadas duas configurações para a VM KVM: com e sem o Virtio. Além disso, foram usados segmentos TCP de 64 bytes para que o processamento da máquina geradora fosse sobrecarregado. O tráfego de pacotes durou 10s. Quinze baterias de teste, assumindo um intervalo de confiança de 95%, foram realizadas e as médias das taxas de encaminhamento de pacotes são apresentadas na Fig. 4. Em todos os testes a taxa de transmissão foi medida na máquina geradora de tráfego, onde o cliente do Iperf foi executado, e a taxa de recepção foi medida na máquina receptora, onde o servidor do Iperf foi executado.

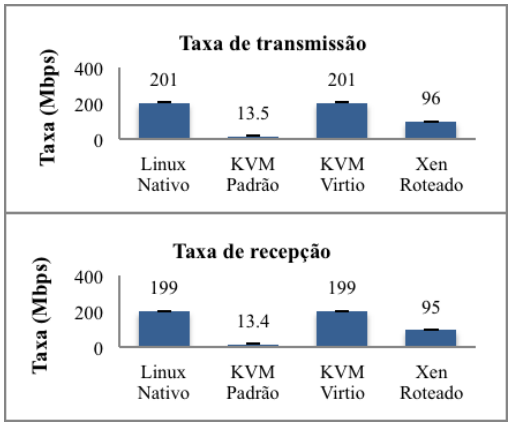

Fig. 4. Taxas para pacotes de 64 bytes na topologia ponto-a-ponto.

Os resultados mostram que o desempenho de encaminhamento do KVM é baixo quando o SO virtualizado executa as operações de E/S de rede no modo de usuário do SO hospedeiro, sem saber que sua interface de rede é virtual (KVM Padrão). Ou seja, o elevado custo da emulação dos dispositivos de rede, feita pelo QEMU, torna o desempenho do encaminhamento de pacotes do KVM muito inferior ao do Xen. Porém, quando a biblioteca libvirt é utilizada para que as operações de E/S sejam virtualizadas [7] o desempenho na transmissão e na recepção de pacotes do KVM Virtio foi semelhante ao encontrado no Linux nativo (Fig. 4). Isto aconteceu porque o driver do dispositivo de rede do SO virtualizado passou a realizar as operações de E/S de rede cooperando com o QEMU para acessar o modo de kernel ao realizar o encaminhamento de pacotes, fazendo chamadas privilegiadas para os dispositivos de rede. Com pacotes pequenos, o desempenho do Xen foi muito menor que o do KVM com Virtio e do Linux nativo (Fig. 4). Porém, seu

desempenho foi semelhante ao do KVM com Virtio, e praticamente igual ao do Linux nativo, quando pacotes grandes de 1500 bytes foram utilizados (Fig. 5). Na recepção de pacotes, inclusive, é possível observar que o Xen chegou a apresentar um desempenho um pouco superior ao do KVM com Virtio.

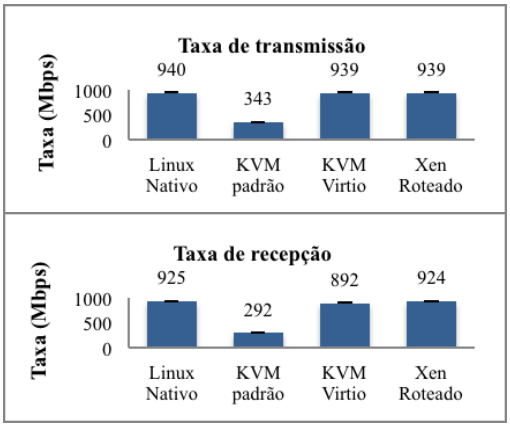

Fig. 5. Taxas para pacotes de 1500 bytes na topologia ponto-a-ponto.

Em seguida, alteramos a topologia para tornar a VM um roteador (Fig. 6), a fim de verificar se o custo de roteamento de pacotes da rede 172.16.50.0/24 (A) para a rede 10.10.50.0/24 (B), alteraria os desempenhos do Xen e do KVM. O mesmo tráfego de pacotes de 1500 bytes foi gerado, do Servidor 3 para o Servidor 1, passando pela VM01 instalada no Servidor 2. Os Servidores 1 e 3 foram instalados com o Linux nativo.

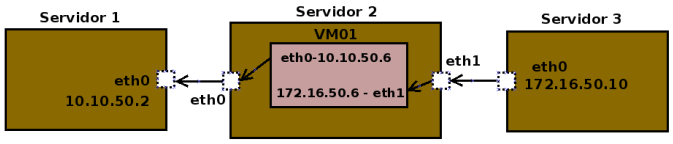

Fig. 6. Topologia do teste de roteamento.

O KVM obteve melhor resultado na transmissão dos dados quando as operações de E/S foram virtualizadas pelo Virtio, e as taxas de recepção mostram que o desempenho do roteador virtual criado no KVM com Virtio é o melhor. O Xen teve resultado pior (Fig. 7) do que os obtidos nos testes realizados com a topologia ponto a ponto (Fig. 5), onde não há tomada de decisão de roteamento e os pacotes são encaminhados em função dos endereços de camada 2 dos dispositivos interligados. Isso ocorreu porque, quando são roteados pela VM Xen, cada pacote precisa atravessar o hipervisor duas vezes. Os pacotes entram pela interface física eth1 do Servidor 2 (Fig. 6) que é visualizada apenas pelo dom0 dessa máquina. Depois de entrar no dom0, os pacotes são direcionados, através do hipervisor para dentro da VM (Figs. 2 e 8), para que a tabela de repasse do roteador virtual seja consultada. Após verificar que a rota de saída aponta para sua interface eth0, o roteador virtual encaminha o pacote de volta para o dom0. Portanto, o pacote volta a atravessar o hipervisor antes de alcançar a interface física eth0. A dupla sobrecarga do hipervisor do Xen torna a taxa de recepção no roteamento de pacotes, de 553 Mbps (na Fig. 7), bem menor que a encontrada quando os pacotes eram apenas encaminhados, de 924 Mbps (Fig. 5).

No KVM, quando um pacote chega na interface física eth1 do Servidor 2 (Fig. 6), ele não precisa atravessar uma VM de drivers para chegar na interface do roteador virtual. O pacote atravessa apenas o hipervisor KVM que está implementado no próprio kernel do SO hospedeiro (Fig. 1). Por esse motivo, o roteamento de um pacote que entrou pela interface física da máquina para a interface de rede da VM, que está declarada dentro de uma árvore de diretórios /dev/kvm exclusiva, é realizado pelo mecanismo de encaminhamento do kernel. A sobrecarga também é menor no KVM quando o pacote precisa deixar a VM para sair pela interface eth0 do Servidor 2 (Fig. 6), desde que o Virtio seja utilizado para permitir que os drivers de rede da VM sejam executados no modo de kernel. Portanto, os resultados obtidos até aqui mostram que o KVM possui melhor desempenho no roteamento de pacotes e exerce menor sobrecarga sobre a CPU e a memória quando comparado com o Xen.

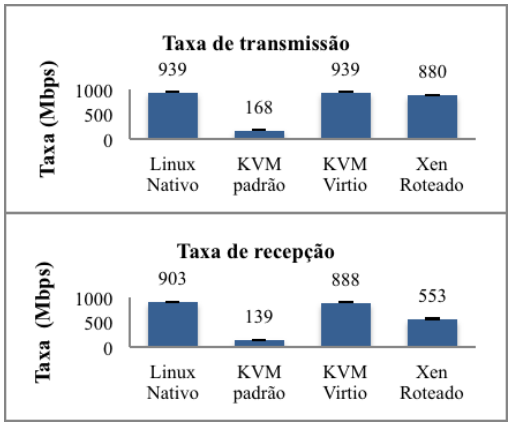

Fig. 7. Taxas para pacotes de 1500 bytes quando a VM é um roteador.

3.3 Capacidade de Roteamento Usando o OpenFlow

Na Seção 3.2, podemos observar o desempenho não satisfatório do Xen quando utilizamos um roteador virtual. Contudo, é possível melhorar seu desempenho no roteamento de pacotes se combinarmos o Xen com o OpenFlow. Quando um comutador OpenFlow é executado no dom0 e controladores OpenFlow virtuais são executados em domUs, os planos de dados das VMs são copiados para o dom0. Isso acontece, porque apenas os primeiros pacotes de um fluxo de dados que entram pela interface eth0 da máquina irão atravessar os domUs, pois uma vez definidas as rotas a serem seguidas, as instruções necessárias para o roteamento são escritas nas tabelas de repasse contidas no domínio de drivers dom0. Portanto, depois que as tabelas de repasse dos controladores domUs são copiadas para o comutador OpenFlow (dom0), todos os pacotes passam a ser roteados apenas pelo dom0, como mostrado na Fig. 8. Essa abordagem é conhecida como separação de planos [8].

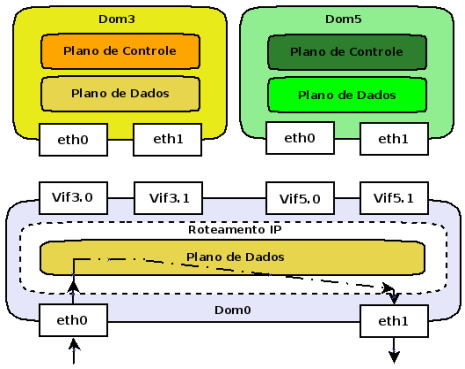

Fig. 8. Exemplo de repasse com separação de planos.

Considere que as máquinas de uma rede utilizam o dom3 como gateway e utilizam a interface eth0 de dom0 para se comunicar com ele. Todos os pacotes enviados desta máquina para outras, que pertencem a redes disponíveis apenas através

da interface eth1 de dom0, precisarão atravessar o gateway dom3 quando não existe a separação de planos (Fig. 9). Os testes de roteamento realizados na Seção anterior foram refeitos com o Xen e o OpenFlow operando em conjunto para tornar o dom0 um plano de dados compartilhado [3]. Também combinamos o KVM com o OpenFlow para fazer esses testes. Porém, nesse caso, tornamos o Linux nativo um comutador OpenFlow. Nos dois casos, o controlador foi configurado dentro das VMs Xen e KVM e 15 rodadas de testes foram executadas assumindo-se um intervalo de confiança de 95%.

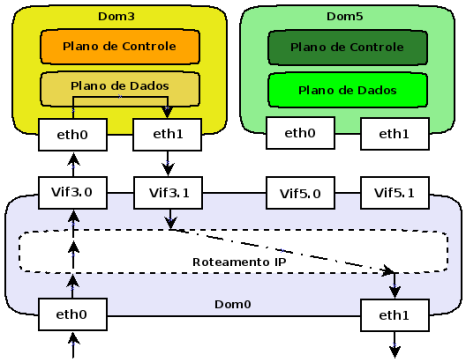

Fig. 9. Exemplo de repasse sem separação de planos.

O desempenho foi melhor (Fig. 10) quando o paradigma da separação de planos foi utilizado (Fig. 8), porque os pacotes deixaram de atravessar o hipervisor duas vezes como acontece na configuração padrão do Xen roteado (Fig. 9). Sem a dupla sobrecarga do hipervisor, o dom0 utilizou as ferramentas de roteamento do kernel, da mesma forma que acontece no Linux nativo, e a eficiência do Xen no roteamento de pacotes melhorou significativamente, tornando-se praticamente semelhante a do Linux nativo (Fig. 10). Os desempenhos de transmissão e recepção dos pacotes roteados praticamente se igualaram ao do Linux nativo também quando o KVM e o OpenFlow foram combinados. Neste caso, o roteamento propriamente dito foi realizado pelo comutador OpenFlow – que no KVM é o próprio Linux e no Xen é o dom0.

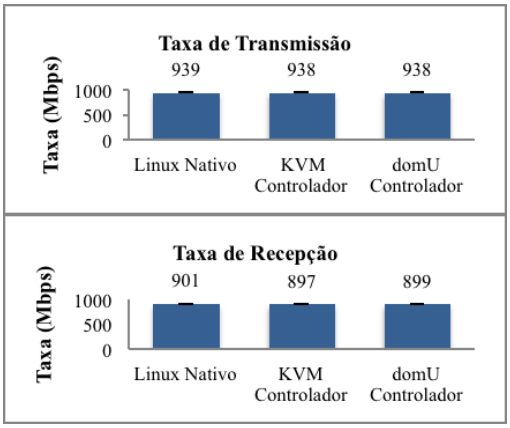

Fig. 10. Taxas com repasse sem separação de planos.

Portanto, os resultados mostram que é possível igualar o desempenho de roteamento das plataformas de virtualização Xen e KVM ao do Linux nativo quando elas são combinadas com o OpenFlow em uma solução de virtualização. Além disso, também é possível observar que o uso de controladores virtuais permite a extensibilidade do plano de dados. Ou seja, caso um controlador precise ser migrado de um nó físico para outro, é possível fazer isso sem interromper o funcionamento dos fluxos em operação na rede virtual, porque os pacotes continuam sendo roteados pelos comutadores que possuem uma cópia do plano de dados do controlador.

IV. CONCLUSÕES

Nos testes realizados, a sobrecarga de memória e CPU do hipervisor do KVM foi menor que a do hipervisor do Xen. Mesmo sendo uma ferramenta de virtualização completa, o KVM, obteve desempenho de roteamento de pacotes melhor que o do Xen para-virtualizado, quando foi configurado para utilizar o Virtio porque os drivers de rede das VMs puderam realizar as operações de E/S com os privilégios de núcleo (modo de núcleo ou kernel) do Linux nativo, utilizando seu mecanismo de encaminhamento do kernel. Além disso, o desempenho do Xen foi pior que o do KVM quando os pacotes de 1500 bytes foram roteados porque eles precisaram atravessar o hipervisor duas vezes, duplicando a sobrecarga do Xen. E uma vez que roteadores virtuais precisam possuir mais de uma interface de rede para serem capazes de rotear pacotes, o resultado dos testes de roteamento é mais importante que o obtido quando ocorre apenas encaminhamento de pacotes. Acreditamos que a plataforma de virtualização KVM é a mais adequada porque apresentou melhor desempenho no consumo de memória, de CPU e no roteamento de pacotes quando utilizou o Virtio. Contudo, esse trabalho mostrou que é possível aproximar a capacidade de roteamento de pacotes do Xen, e do KVM, ao do Linux nativo de uma máquina x86, quando essas plataformas de virtualização são combinadas com o OpenFlow para realizarem separação de planos.

REFERÊNCIAS

[1] S. Das, G. Parulkar, N. Mckeown, *Unifying packet and circuit switched networks*. OPENFLOW-TR, Department of Electrical Engineering, Stanford University, 2009.

[2] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, L. Mathy, T. Schooley, *Evaluating Xen for router virtualization*. In Proc. of ICCCN 07. p. 1256-1261, 2007.

[3] N. C. Fernandes, O. C. M. B. Duarte, *XNetMon: A network monitor for securing virtual networks*. In Proceedings of the IEEE International Conference on Communications, ICC 2011.

[4] Gartner, *Magic Quadrant for x86 server virtualization infrastructure*. http://www.gartner.com/technology/reprints.do?id=1-1AVRX JO&ct =120612&st=sb, janeiro 2013.

[5] N. Gude, T. Koponen, J. Pettit, B. Pfaff, *NOX: Towards an operating system for networks*, ACM SIGCOMM Computer Communication Review, Vol. 38, Issue 3, 2008.

[6] Kvm.org, *Preparing to use KVM*, http://www.linux-kvm.org/page/faq, novembro de 2012.

[7] Kvm, *Using Virtio_net for the guest nic: throughput tests using Iperf*, http://www.linux-kvm.org/page/using_virtio_nic, maio de 2013.

[8] D. M. F. Mattos, L. H. G. Ferraz, L. H. M. K. Costa, O. C. M. B. Duarte, *Evaluating virtual router performance for a pluralist future Interne*, In 3th International Conference On Information and Communication Systems, Jordânia, 2012.

[9] L. A. F. Mauricio, M. G. Rubinstein, "Avaliação de desempenho de plataformas de virtualização de redes". Dissertação de Mestrado, PEL/UERJ, Rio de Janeiro, Brasil, 2013.

[10] OpenMP, "Apostila: Introdução ao OpenMP", Unicamp, São Paulo, Brasil, 2010.

[11] J. Rexford, C. Dovrolis, *Future Internet architecture: clean slate versus evolutionary research*, In Communications of the ACM, Vol. 53, n. 9, p. 36-40, 2010.

[12] Stream, *Notes on the mystery of hardware cache performance counters* http://blogs.utexas.edu/jdm4372/, janeiro de 2013.

[13] Super Pi, *A single threaded benchmark that calculates Pi to a specific number of digits*, http://www.superpi.net/, janeiro de 2013.

[14] J. S. Turner, D. E. Taylor, *Diversifying the Internet*, In Proc. IEEE Globecom, 2005.