Análise Multiparamétrica do *Overhead* de Rede em Máquinas Virtuais

Flávio Urschei[1], Éder José Pelegrini[2], Márcio Augusto de Lima e Silva[1], Edson Toshimi Midorikawa[3], Tereza Cristina Melo de Brito Carvalho[1]

[1] Laboratório de Arquitetura e Redes de Computadores

[2] Laboratório de Linguagens e Técnicas Adaptativas

[3]Laboratório de Arquitetura e Computação de Alto Desempenho

Departamento Engenharia de Computação e Sistemas Digitais – Escola Politécnica
Universidade de São Paulo – São Paulo – Brasil

{furschei, carvalho, msilva}@larc.usp.br,
{eder.pelegrini,edson.midorikawa}@poli.usp.br

Abstract. *This work assesses the overhead introduced by virtualization layer in networking performance, specifically evaluating the performance of the XEN, one of the largely used virtual machine monitors. The metric used is the throughput (in Mbps), measured with the netperf benchmark. The parameters network MTU, application buffer size and system buffer size were varied in order to verify their influence over the system throughput.*

Resumo. *O objetivo deste trabalho é avaliar a influência do overhead introduzido pela camada de virtualização no desempenho de rede. Especificamente, irá analisar o desempenho do XEN, um dos monitores de máquinas virtuais amplamente utilizados. A métrica utilizada é a vazão (em Mbps) obtida em comunicações TCP/IP, medida com o benchmark netperf. Os parâmetros MTU da rede, tamanho do buffer da aplicação e tamanho do buffer de sistema (tamanho da janela TCP) foram variados de modo a verificar a influência dos mesmos na vazão do sistema.*

1 Introdução

A virtualização é uma importante técnica para o projeto de sistemas computacionais, visto que é utilizada por diversos sistemas com o objetivo principal de desacoplar os recursos físicos de um determinado *hardware* da sua representação lógica a um usuário externo. Para este fim é introduzida uma camada de abstração, a qual permite a criação de múltiplas instâncias lógicas, cada qual de posse de uma fração da totalidade dos recursos de uma instância física, ou ainda, a criação de uma única instância lógica a partir de múltiplas instâncias físicas [Smith e Nair, 2005] [Barham, Dragovic *et al.*, 2003].

Quando um sistema computacional de propósito geral é virtualizado, é necessário um componente de gerenciamento, denominado monitor de máquinas virtuais (MMV), o qual consiste de uma camada de software que provê uma interface entre os recursos reais e as máquinas virtuais, fornecendo uma abstração de tais recursos

para as mesmas [Rose, 2004]. Essencialmente o MMV virtualiza a CPU, o sistema de gerenciamento de memória e o sistema de entrada/saída (E/S).

Aplicações do tipo *CPU bound*, que realizam poucas operações de E/S, são pouco penalizadas pela camada de virtualização, ao passo que aquelas que são do tipo *I/O bound* realizam muitas trocas de contexto entre o espaço do sistema hóspede e o espaço do MMV, implicando em uma maior penalidade de desempenho [Barham, Dragovic *et al.*, 2003]. Desse modo, dentre os componentes virtualizados (CPU, memória e E/S), o que gera maior *overhead* de virtualização são as operações de E/S, em razão da complexidade dos protocolos de entrada e saída, i.e. uma requisição de E/S deve ser tratada por múltiplas camadas de protocolos existentes entre o espaço de usuário e o hardware do sistema. Por exemplo, em uma transmissão TCP, o segmento irá passar pela pilha de protocolos do sistema operacional hóspede até atingir a interface virtual com o MMV. A partir daí, o MMV deve fazer o correto interfaceamento entre as interfaces virtuais (dos hóspedes) e a(s) interface(s) física(s), o que ocasiona um *overhead* de transmissão.

O objetivo deste trabalho é avaliar a influência do *overhead* introduzido pela camada de virtualização no desempenho de rede. Especificamente, irá analisar o desempenho de um dos MMVs amplamente utilizados: o XEN [Xen, 2006].

As próximas seções do artigo organizam-se como segue: na seção 2 são discutidos trabalhos relacionados e na seção 3 são descritos aspectos conceituais dos mecanismos de virtualização usados pelo XEN, focando principalmente em entrada e saída. A seção 4 é dedicada à avaliação do *overhead* de rede, e apresenta a configuração dos experimentos, seguido dos resultados e análise dos mesmos. Por fim, as conclusões e trabalhos futuros estão na seção 5.

2 Trabalhos Relacionados

Há diversos trabalhos publicados referentes à avaliação de desempenho de máquinas virtuais. Alguns como o apresentado em [Ahmad, Anderson *et al.*, 2003] analisam aspectos referentes ao desempenho de disco, outros como em [Whitaker, Shaw *et al.*, 2002] apresentam uma análise do desempenho global do sistema virtualizado incluindo aspectos referentes ao disco, rede, CPU e memória. Existem aqueles que, apesar de versarem sobre o desempenho de máquinas virtuais, são focados em uma ferramenta de instrumentação específica, como em [Menon, Santos *et al.*, 2005] em que é apresentada a ferramenta Xenoprof, baseada no Oprofile [Oprofile, 2006], que permite determinar a distribuição de eventos de hardware como ciclos de *clock*, taxa de erros em *caches* e em TLBs. Há ainda trabalhos como o apresentado em [Huang, Liu *et al.*, 2006] que analisam o desempenho de máquinas virtuais aplicado a Computação de Alto Desempenho. São poucos os artigos que versam única e especificamente sobre o desempenho de rede em máquinas virtuais. A seguir são apresentados dois desses trabalhos que foram considerados relevantes para a elaboração desse artigo.

Em [Sugerman, Venkitachalam *et al.*, 2001], é apresentado um estudo detalhado sobre a virtualização de dispositivos de entrada e saída para o VMware Workstation, o qual é um monitor de máquinas virtuais que executa como uma aplicação sobre um outro sistema operacional (*host*). Este artigo foca suas análises no desempenho de adaptadores Ethernet, e sugere otimizações como modificação do sistema operacional

hóspede e otimização do *driver* de comunicação do mesmo. Apresenta ainda um método de evitar a manipulação de chamadas pelo sistema operacional hospedeiro (*bypassing the host OS*). Apesar deste artigo considerar a influência do tamanho do *buffer* de aplicação (*data size*) no desempenho de rede, não considera o buffer de sistema (*socket size*) e não realiza uma análise extensiva sobre a influência desses parâmetros.

Em [Barham, Dragovic *et al.*, 2003] é apresentado um estudo sobre o XEN. Nesse artigo são descritas as técnicas de virtualização usadas nesse monitor de máquinas virtuais, o qual utiliza o conceito de paravirtualização e anéis de descritores para a realização de entrada e saída. Este trabalho também apresenta uma análise comparativa de desempenho entre o XEN, o VMware, o *User Mode Linux* e o Linux nativo, por meio de uma série de *benchmarks* (SPEC INT 2000, OSDB, SPEC WEB99, entre outros). Dentre os *benchmarks* utilizados, consta o `ttcp`, empregado para avaliar o desempenho de comunicações TCP. Foi considerado um tamanho de *buffer* de sistema fixo (128KB) e dois valores para o MTU (*Maximum Transmit Unit*) da rede (500 e 1500 bytes), no entanto, essa consideração despreza a influência de outros parâmetros, como o tamanho do *buffer* de aplicação, além da própria variação do tamanho do *buffer* do sistema, os quais têm influência significativa no desempenho de comunicações TCP.

Considerando os trabalhos mencionados, este artigo procura complementar os demais por meio da análise multiparamétrica do desempenho de rede em ambientes virtualizados, especificamente para o MMV XEN [1].

3 Virtualização de Entrada e Saída no XEN

O XEN [Barham, Dragovic *et al.*, 2003] é um monitor de máquinas virtuais para plataformas x86, desenvolvido pela Universidade de Cambridge, o qual permite que múltiplos sistemas operacionais compartilhem recursos de hardware garantindo funcionalidade e desempenho. Para contornar o problema da não virtualização da arquitetura IA-32 [Robin e Irvine, 2000], utiliza a abordagem de paravirtualização em contrapartida à virtualização completa [Rose, 2004].

Na paravirtualização, o *kernel* do sistema operacional da máquina hóspede (domínio 1 ou dom1 na terminologia do Xen) é modificado de modo que o MMV (denominado *hypervisor* para o caso do XEN) execute tarefas protegidas (i.e. não sejam direcionadas diretamente para a CPU). No XEN, o *hypervisor* está executando no nível 0 (*ring 0*), de modo que apenas ele possa acessar o hardware diretamente. Os *kernels* dos sistemas operacionais hóspedes são modificados para executarem no nível 1 (*ring 1*), isto é, as máquinas virtuais não são capazes de acionar diretamente o hardware. As aplicações nos sistemas operacionais hóspedes continuam a executar no nível 3 (*ring 3*). Nessa abordagem, o escalonamento das máquinas virtuais no hardware é realizado pelo *hypervisor*. Devido à modificação das interfaces dos *kernels* nos hóspedes, a máquina física necessita de um *kernel* modificado (domínio 0 ou dom0).

A presença do *hypervisor* se coloca como um domínio adicional de proteção entre os sistemas operacionais hóspedes (dom1) e os dispositivos de entrada e saída. Dessa forma, é fundamental um mecanismo de transferência de dados que permita a

[1] Foram realizados estudos comparativos entre o XEN e VMware Server, no entanto, devido a restrições impostas pela VMware na EULA, os resultados relativos ao VMware Server não podem ser publicados.

movimentação de dados verticalmente pelo sistema com um baixo *overhead* [Barham, Dragovic *et al.*, 2003]. O XEN não emula dispositivos de hardware existentes como é feito nos ambientes com virtualização completa. Dados de entrada e saída são transferidos para (e de) cada domínio via XEN, por meio de memória compartilhada e por meio de anéis de descritores assíncronos. Um anel é uma fila circular de descritores alocados por um domínio, mas acessados internamente ao XEN. Os descritores não contêm dados de entrada e saída diretamente, ou seja, *buffers* de dados de E/S são alocados de forma separada pelos sistemas operacionais hóspedes e são diretamente referenciados pelos descritores de entrada e saída. Dessa forma não é necessário mapear ou copiar os dados no espaço de endereçamento do XEN, ficando confinados às áreas dos sistemas operacionais hóspedes. Descritores são pequenos e exigem pouco processamento para cópia e validação. Os mesmos sempre se referem a massas de dados. O acesso a cada anel está baseado em dois pares de ponteiros produtor-consumidor. Os domínios arranjam as requisições de E/S em um anel por meio do avanço do ponteiro de requisição de consumo associado. As respostas às requisições de E/S são arranjadas no anel de forma similar, mas agora com o XEN sendo o produtor e com o sistema operacional hóspede sendo o consumidor. É importante ressaltar que não há requisitos para que as requisições de E/S sejam processadas em ordem, visto que os sistemas operacionais hóspedes associam um identificador único com cada requisição de E/S, o qual é reproduzido na resposta associada, permitindo assim reordenar as operações de E/S conforme considerações de escalonamento e prioridade. A estrutura de anéis do XEN é suficientemente genérica para suportar diversos tipos de dispositivos [Barham, Dragovic *et al.*, 2003], sendo que acessos ao disco e a rede utilizam essa técnica.

Para o caso específico de rede, há dois anéis: um para transmissão e outro para recepção. Para transmitir um pacote, o sistema operacional hóspede simplesmente enfileira um descritor de *buffer* no anel de transmissão. O XEN copia o descritor e, por questões de segurança (*safety*), copia o cabeçalho (o *payload* não é copiado) do pacote, executando eventuais ações de filtragem. O XEN implementa a política de *round-robin* para escalonar os pacotes para as diferentes máquinas virtuais. Na recepção de pacotes, o hóspede troca um *frame* de página não usado para cada pacote que ele recebe, evitando a necessidade de copiar o pacote entre o XEN e o sistema operacional hóspede. Quando um pacote é recebido, o XEN imediatamente encaminha o mesmo para a interface virtual (VIF) do hóspede de destino, e troca o *buffer* do pacote pelo *frame* de página no correspondente anel de recepção do sistema operacional hóspede (destino do pacote).

O XEN implementa uma *bridge* virtual, ou seja, os *frames* são encaminhados apenas para as interfaces de destino, não sendo realizado *broadcast* para todas as interfaces virtuais. A Figura 1 apresenta a estrutura de rede virtual para o XEN. Nessa estrutura, a *bridge* repassa os *frames* que chegam por uma interface para a respectiva interface de destino. O XEN renomeia as interfaces na inicialização: a interface `eth0` (física) é renomeada para `peth0`, o endereço MAC da interface física real é copiado para a veth0, a qual é renomeada para eth0. Internamente, a interface `veth0` é vinculada à interface virtual `vif0.0` que se conecta à *bridge* (`xenbr0`). Analogamente, uma máquina virtual `X` tem cada interface `ethY` vinculada a uma interface virtual `vifX.Y`, cada qual está conectada a `xenbr0`.

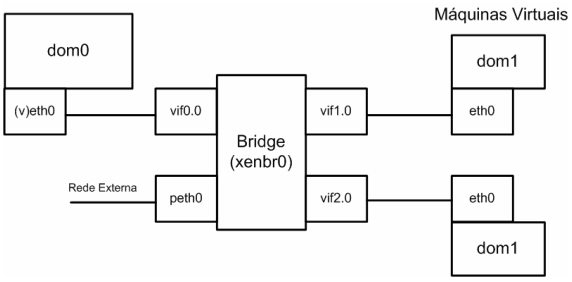


Figura 1. Estrutura de rede virtual do XEN.

4 Desempenho de rede de máquinas virtuais

Conforme fora supracitado, o objetivo desse trabalho é analisar o *overhead* introduzido pela camada de virtualização do XEN no desempenho de rede. A métrica utilizada para avaliar e comparar o desempenho de rede das máquinas virtuais estudadas foi a vazão da rede em Mbps. Esta escolha se deve ao fato de que todas as operações necessárias à transmissão de uma unidade de dados de um ponto a outro, e realizadas pelos diferentes componentes do sistema computacional (monitor de máquinas virtuais, sistema operacional hóspede, sistema operacional hospedeiro, entre outros) impactam diretamente na vazão da rede. Desse modo todo o *overhead* introduzido pela virtualização irá refletir diretamente na vazão.

Em uma comunicação TCP, existem diversos parâmetros que podem influenciar na vazão, dentre os quais tamanho do *buffer* da aplicação (*message size*), tamanho do *buffer* de sistema (*socket size* ou *window size*) e MTU. Em relação ao *buffer* da aplicação, valores pequenos fazem com que sejam necessárias muitas operações de escrita da área de aplicação para o *buffer* do sistema para que uma transmissão ocorra, sendo assim um aumento do *buffer* de aplicação impacta positivamente na vazão. Em relação ao *buffer* de sistema, quanto maior seu tamanho, menor será a quantidade de mensagens de confirmação enviadas pelo TCP, o que conduz a um aumento da vazão. Cumpre ressaltar que quando o tamanho do *buffer* da aplicação é igual ao tamanho do *buffer* de sistema, o desempenho tende a ser melhor, visto que as transmissões iniciar-se-ão imediatamente após uma escrita do *buffer* da aplicação no de sistema, sendo necessária apenas uma escrita para iniciar o envio da mensagem. Por fim, em relação ao MTU, valores pequenos impactam negativamente na vazão da rede, visto que será necessária uma quantidade maior de mensagens e conseqüentemente uma maior quantidade de requisições de entrada e saída será necessária, aumentando o *overhead* de transmissão. Tendo em vista essa análise qualitativa, os experimentos descritos nas próximas seções visam verificar como esses parâmetros impactam no desempenho da rede diante das diferentes técnicas de virtualização.

O experimento realizado nesse trabalho tem como objetivo medir o *overhead* decorrente da virtualização na comunicação entre diferentes máquinas físicas em função dos diferentes parâmetros de rede mencionados. Nesse estudo foi utilizado o *benchmark* `netperf` [Netperf, 2006], o qual permite que sejam variados o tamanho do *buffer* da aplicação (*message size* – opção `-m`), e o tamanho do *buffer* de sistema (*socket size* – opção `-s`). As iterações dentro de cada caso de teste foram executadas por no mínimo 21s, correspondentes a 3 medidas, e por no máximo 70s, correspondentes a 10 medidas,

encerrando a execução de uma iteração assim que for atingido um valor dentro do intervalo de confiança de 95%.

4.1 Configuração do Ambiente Experimental

O experimento foi realizado usando dois computadores conectados fisicamente por meio de placas Gigabit Ethernet, usando um cabo cruzado. Ambos os computadores possuem hardware idêntico (Pentium D 3 GHz, 2GB de memória RAM). Foi utilizado o sistema operacional Linux (Fedora Core 5 – FC5) como sistema hóspede. Para o caso do hospedeiro, foi utilizada a versão modificada deste mesmo sistema operacional (*kernel* Xen0 para o domínio 0 e *kernel* XenU para o domínio 1), executando sobre o XEN 3.0 [Xen, 2006].

4.2 Overhead de comunicação em sistemas virtualizados

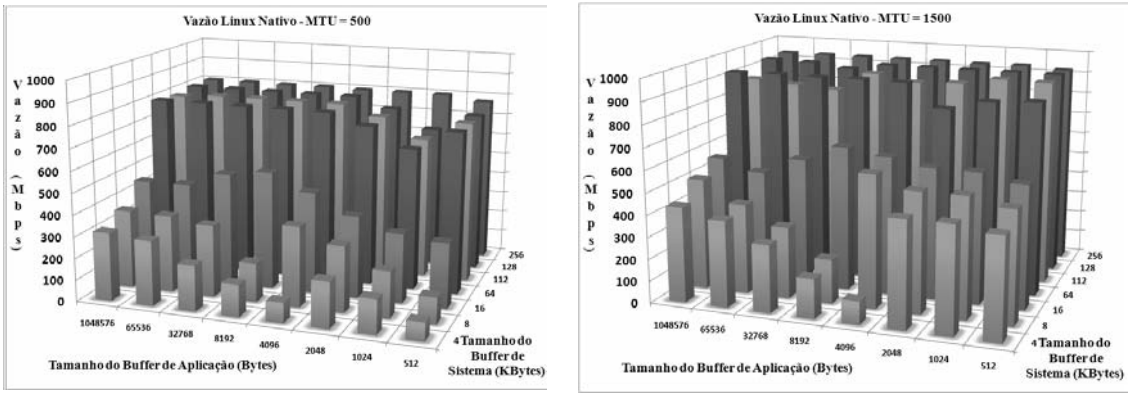
O objetivo desse experimento é medir a vazão da rede entre uma máquina virtual hospedada em uma máquina física M1 comunicando-se diretamente com uma segunda máquina física M2. A máquina M2 sempre está configurada com Linux nativo, ao passo que a configuração da máquina M1 varia de acordo com o descrito na Tabela 1.

Tabela 1. Configurações experimentais: máquina real – máquina virtual

Config.	Máquina 1 (M1)	Objetivo
1	Linux Nativo	Estabelecer uma medida de referência para a vazão da rede, em função dos 3 parâmetros considerados.
2	Dom0 do XEN	Avaliar a influência do *kernel* modificado (Xen0) na vazão da rede, ou seja, o quanto a vazão varia em relação à primeira configuração.
3	Dom1 do XEN (Xen1)	Avaliar a influência do XEN *hypervisor* com uma máquina virtual (Xen1) na vazão da rede, ou seja, o quanto o *overhead* de virtualização influencia no desempenho.

Para cada uma das configurações, foram realizadas medidas para MTU igual a 500 e 1500 bytes. Para cada valor de MTU foram escolhidos os seguintes valores para *buffer* do sistema: 4096, 8192, 16384, 65536, 114688, 131072 e 262144 bytes. Por sua vez, para cada valor de *buffer* do sistema foram escolhidos os seguintes valores para o *buffer* da aplicação: 512, 1024, 2048, 4096, 8192, 32768, 65536 e 1048576 bytes.

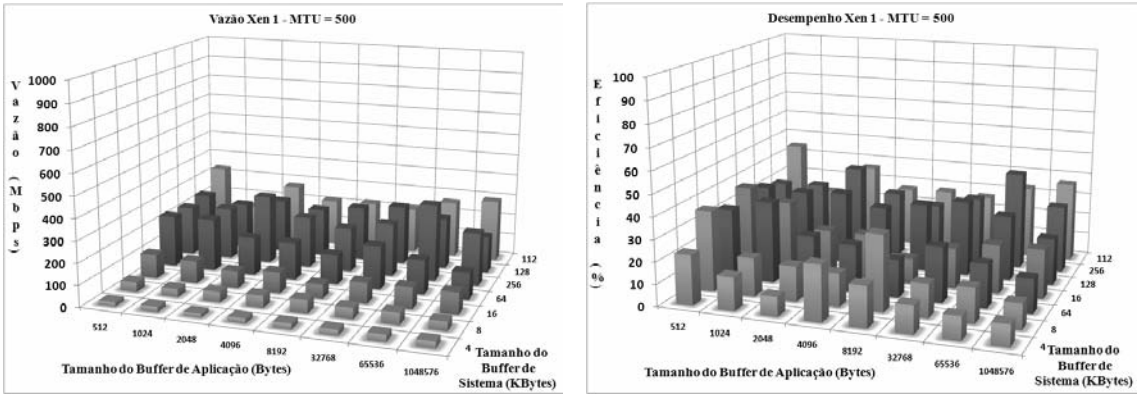
Os resultados das medições são apresentados nos gráficos das Figuras 2 a 7. Cumpre ressaltar que nos gráficos apresentados nestas figuras os valores de *buffer* de sistema e *buffer* de aplicação estão organizados de modo a otimizar a visualização. A Figura 2 apresenta os valores de vazão absolutos (em Mbps) para as medidas da configuração 1, que servem como referência para a determinação da eficiência da estrutura de virtualização de rede do Xen. Destacam-se os seguintes comportamentos dos dados: (i) queda de performance mediante a redução do MTU; (ii) com o aumento do tamanho do *buffer* de sistema, a vazão tende a aumentar, aproximando do limite teórico para os maiores valores de *buffer* de sistema utilizados; (iii) para o mesmo valor de MTU e *buffer* de sistema de 112, 128, 256 KBytes, as vazões obtidas apresentam valores próximos da vazão máxima medida; e (iv) para os valores de 4 e 8 Kbytes, a vazão tende a ter o pior desempenho na situação em que o valor do *buffer* de aplicação é igual ao do *buffer* de sistema.



(a) (b)

Figura 2. Vazões obtidas com o Linux Nativo – config. 1. (a) MTU 500. (b) MTU 1500.

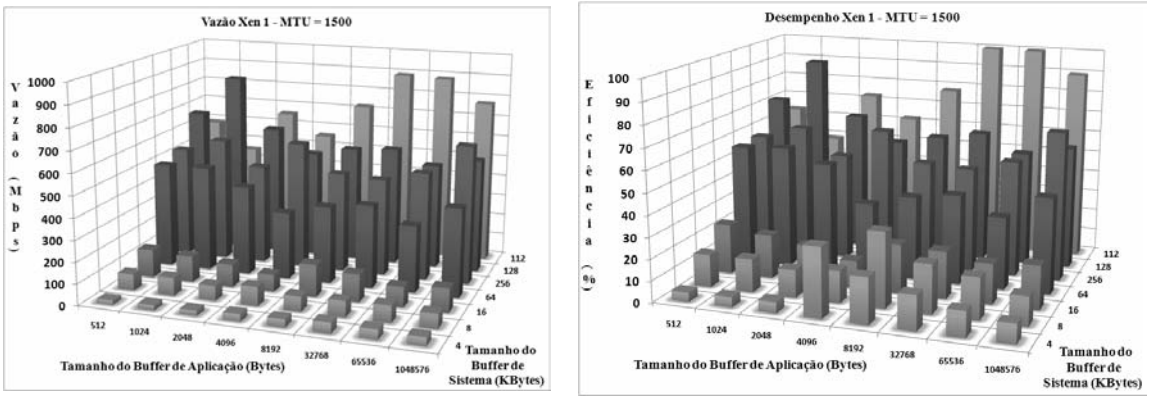
A Figura 3 apresenta a vazão (a) e os valores percentuais (eficiência) da relação entre a vazão medida nas máquinas virtuais contra a vazão medida no Linux nativo (b) para os parâmetros considerados. A análise dos dados aponta para: (i) uma queda na eficiência de no mínimo 50%; (ii) os valores de *buffer* de sistema 112, 128 e 256 Kbytes representam os melhores pontos de operação, tanto para a vazão absoluta quanto para a eficiência; e (iii) a sensibilidade do desempenho de rede em função da variabilidade dos *buffers*.



(a) (b)

Figura 3. Resultados da máquina virtual Xen (domínio 1) para MTU 500. (a) Vazão. (b) Eficiência relativa à máquina nativa.

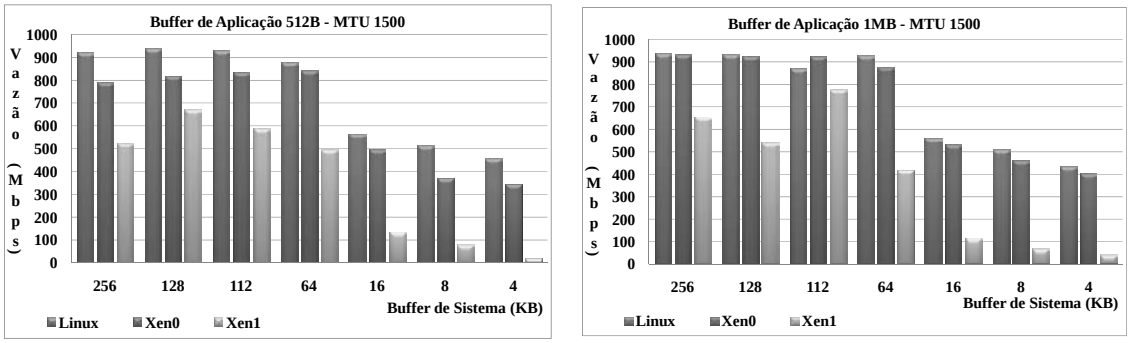
A Figura 4 é análoga à Figura 3, porém para o valor de MTU igual a 1500 bytes. Os dados indicam: (i) eficiência do XEN em torno de 100% para *buffer* de sistema de 112KB e *buffer* de aplicação de 32 e 64KB; (ii) eficiência e vazão média superior ao caso anterior para MTU 500; e (iii) a sensibilidade do desempenho de rede em função da variabilidade do MTU e *buffers*.



(a) (b)

Figura 4. Resultados da máquina virtual Xen (domínio 1) para MTU 1500. (a) Vazão. (b) Eficiência relativa à máquina nativa.

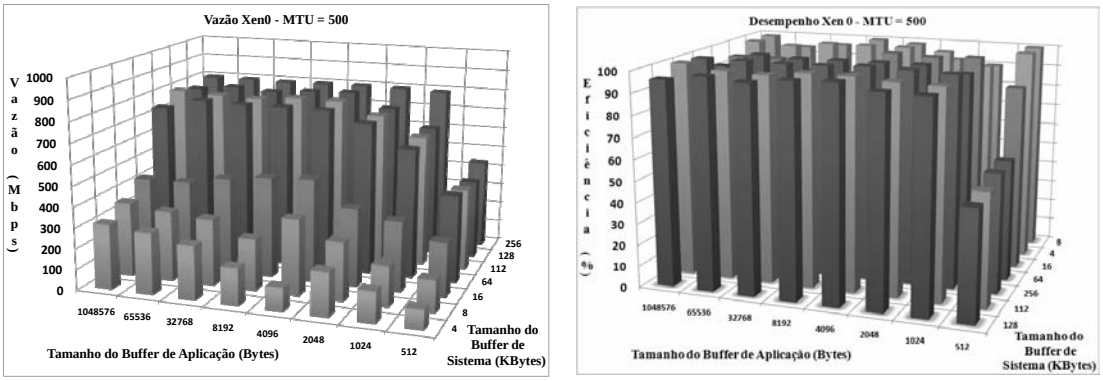
A Figura 5 apresenta as vazões absolutas medidas nas 3 diferentes configurações para os valores extremos (512B e 1MB) de *buffer* de aplicação. Observam-se: (i) vazões próximas entre Linux nativo e Xen0 (dom0) independentemente da variabilidade dos *buffers*; e (ii) eficiência média superior do domínio 0 do Xen com relação ao domínio 1, com destaque para valores de *buffer* de sistema iguais a 16, 8 e 4 Kbytes, onde a diferença supera 300%;



(a) (b)

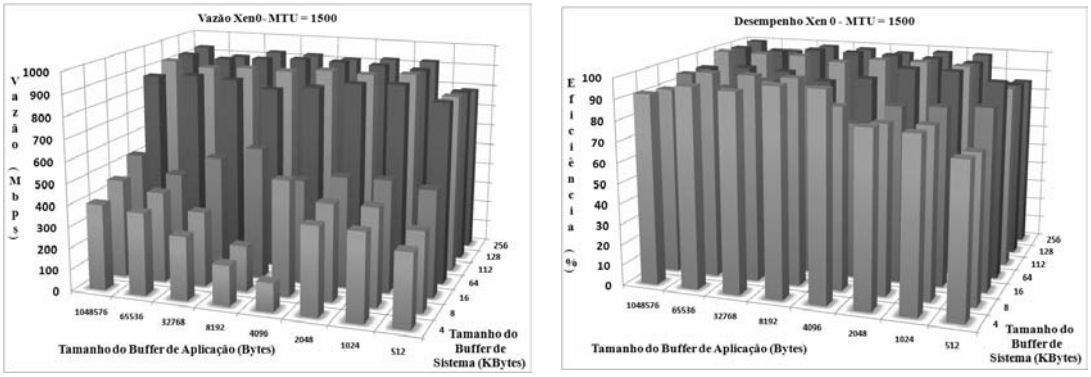
Figura 5. Vazão absoluta para MTU 1500. (a) Buffer de aplicação de 512B. (b) Buffer de aplicação de 1MB.

A análise dos dados numéricos apresentados indica que, com relação à configuração 2, o domínio 0 do Xen apresentou resultados ligeiramente menores que o Linux nativo, com exceção para MTU igual a 500, *buffer* de aplicação igual a 512 bytes e *buffer* de sistema variando de 64KB a 256 KB, onde a vazão para o caso do domínio 0 apresentou uma queda em torno de 50% com relação ao Linux nativo – Figura 6 e 7. Na maioria dos casos, a ligeira queda de desempenho com relação ao Linux nativo deve-se ao fato do domínio 0 também utilizar a estrutura de rede virtual do *hypervisor* Xen (*bridged*), a qual introduz um *overhead*. Os altos índices de desempenho e a divergência de valores entre o Xen0 e o Xen1 indicam que, apesar do domínio 0, aparentemente, ter acesso às estruturas virtuais de rede, este domínio faz uso das rotinas de rede do Linux hospedeiro e, portanto, não utiliza o anel de descritores assíncronos, o que introduz um menor *overhead*.



(a) (b)

Figura 6. Resultados do dom0 (Xen0) para MTU 500. (a) Vazão. (b) Eficiência relativa à máquina nativa.



(a) (b)

Figura 7. Resultados do dom0 (Xen0) para MTU 1500. (a) Vazão. (b) Eficiência relativa à máquina nativa.

Com relação à configuração 3, foi verificada a sensibilidade da eficiência em função dos três parâmetros de redes estudados. A variação do tamanho do *buffer* de aplicação e do *buffer* de sistema afeta a freqüência de troca de contextos entre o espaço de usuário e do sistema operacional hóspede, implicando em um *overhead* maior comparado ao caso do Linux nativo.

Para o caso do *buffer* de sistema, existe o problema do *overhead* relacionado ao sub-aproveitamento da capacidade de transmissão da rede. Caso o *buffer* de sistema não seja múltiplo do MTU, ao se enviar uma unidade de dados equivalente ao *buffer* do sistema (o envio dos dados do *buffer* de sistema ocorre quando o mesmo enche com os dados do *buffer* de aplicação), o último *frame* relativo ao *buffer* de sistema enviado possuirá um tamanho menor do que o MTU. Nessa situação, estas mensagens possuem um comportamento similar ao caso de redução do MTU, onde cada descritor da estrutura de anel de descritores assíncrona representa uma menor quantidade de dados, ou seja, o *overhead* de criação e tratamento do descritor não varia com a quantidade de dados. Entretanto, ao representar uma menor quantidade de dados, o desempenho de rede tende a cair. Ao comparar o tamanho de *buffer* com ordens de grandezas diferentes (como 4 Kbytes e 112 Kbytes) temos que os *buffers* de sistemas maiores conduzem a situações com menores *overheads*. Para um *buffer* de sistema de 4 Kbytes e um MTU de

1500 bytes, a freqüência do *frame* com maior *overhead* é de 1 em 3, enquanto para 112 Kbytes é de 1 em 75 (valores aproximados).

Ao se reduzir o MTU, o XEN apresenta queda significativa na vazão, dado o aumento do número de interrupções que devem ser tratadas, em razão do aumento do número de *frames* necessários para a transferência de uma mesma quantidade de dados, o que também contribui para a sobrecarga das estruturas de virtualização de rede. Mais especificamente, apesar da quantidade de dados efetiva a ser transferida ser menor, a sobrecarga nas estruturas de anéis de descritores pode ser considerada o fator dominante (a movimentação ocorre por meio da transferência do conteúdo de uma página no domínio de origem para outra no domínio de destino).

Outro fator que acarreta na variação do desempenho em função do buffer de aplicação refere-se à operação de desfragmentação dos *buffers* da fila de envio/recebimento de pacotes. Em [Menon, Santos *et al.*, 2005] é investigado o motivo da variação do *overhead* em função do *buffer* de aplicação. Quando a memória da fila de recebimento de pacotes do soquete TCP excede o limite, o Linux invoca uma rotina com o objetivo de liberar memória fragmentada nesta fila com o intuito de evitar perdas. Esta liberação de memória é feita através da cópia dos *buffers* da fila para novos *buffers* compactos (menor tamanho que o anterior). Esta operação de compactação dos *buffers* é uma operação custosa, podendo acarretar considerável *overhead* na comunicação de rede. Tipicamente, o tamanho dos *buffers* que formam a fila é igual ao valor do MTU, o que implica em pouca fragmentação interna, entretanto o Xen aloca uma página de memória (4 Kbytes) para tais *buffers*, ou seja, aloca 4 Kbytes por *frame* recebido independentemente do MTU utilizado. Esta escolha é motivada para facilitar a troca de dados entre domínios, resultando em uma fragmentação interna, o que acarreta queda de desempenho devido à operação de desfragmentação dos *buffers*.

5 Conclusões e Trabalhos Futuros

A contribuição desse primeiro trabalho sobre a análise multiparamétrica do *overhead* de rede em máquinas virtuais refere-se à avaliação do desempenho de rede considerando a introdução da camada de virtualização, em conjunto com a variação de outros parâmetros importantes *(buffer* de sistema e de aplicação e MTU da rede) quando se trata de comunicações de rede por meio do protocolo TCP. A variação desses parâmetros influencia no *overhead* ocasionado pela camada de virtualização, sendo observadas situações em que a vazão da rede aproxima-se, em menor ou maior grau, da vazão entre máquinas nativas. A grande variabilidade encontrada nesse estudo indica que, ao se usar máquinas virtuais em aplicações que exijam um desempenho de rede adequado (ex: utilizar máquinas virtuais para implementar roteadores por meio de software, ou para isolar aplicações como servidores Web), deve-se escolher adequadamente as configurações de outros parâmetros (tamanho de *buffers* e MTU) de modo a maximizar o desempenho da aplicação envolvida.

De modo a melhor explicar os comportamentos observados, propõem-se alguns trabalhos futuros. O primeiro deles consiste em repetir os experimentos monitorando outras variáveis de sistema, como carga na CPU, número de faltas de páginas e número de trocas de contexto. Para isso é necessário usar uma ferramenta adequada para instrumentação de *kernel*. O segundo consiste em realizar os experimentos para o protocolo UDP, não orientado a conexão, o qual não necessita de confirmações. Um

terceiro experimento consiste em testar as estruturas de virtualização de rede dos monitores de máquinas virtuais por meio de experimentos entre pares de máquinas interconectadas por *Virtual LANs* independentes. Por fim, um quarto experimento consiste em verificar a distribuição de carga das máquinas virtuais na presença de mais de uma interface física. Ou seja, ao se adicionar mais de uma interface real, e se distribuir pelas mesmas o fluxo das diversas *Virtual LANs* que conectam as máquinas virtuais, verificar se ocorre um aumento proporcional da vazão em função do número de interfaces físicas.

Além dos experimentos mencionados que utilizam plataformas x86 não virtualizáveis, outro possível estudo consiste em avaliar o desempenho de rede quando esses monitores de máquinas virtuais residem sobre plataformas virtualizáveis (Intel VT ou AMD-V). Uma outra tecnologia que poderia ser avaliada no que diz respeito à plataformas virtualizáveis é o KVM (*Kernel-based Virtual Machine*) [KVM, 2007], o qual consiste em uma solução completa para virtualização para Linux sobre hardware x86, e faz uso das extensões para virtualização disponíveis em processadores modernos (Intel VT ou AMD-V).

Referências

Ahmad, I., J. M. Anderson, et al. (2003). "An analysis of disk performance in VMware ESX server virtual machines". IEEE International Workshop on Workload Characterization (WWC-6). p. 65 - 76.

Barham, P., B. Dragovic, et al. (2003). "Xen and the Art of Virtualization". 19th ACM Symposium on Operating Systems Principles (SOSP 2003). Bolton Landing, NY, USA: ACM Press. p. 164 -177.

Huang, W., J. Liu, et al. (2006). "A Case for High Performance Computing with Virtual Machines". International Conference on Supercomputing. Cairns, Queensland, Australia: ACM Press. p. 125-134.

KVM (2007). "Kernel-based Virtual Machine". Disponível em http://kvm.qumranet.com/kvmwiki. Acessado em 05/02/2007.

Menon, A., J. R. Santos, et al. (2005). "Diagnosing performance overheads in the xen virtual machine environment". Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments, p.13-23.

Netperf. (2006). Disponível em www.netperf.org. Acessado em 01 Nov. 2006.

Oprofile (2006). Disponível em http://oprofile.sourceforge.net/. Acessado em 07/03/2007.

Robin, J. S. e C. E. Irvine (2000). "Analysis of the Intel Pentium's ability to support a secure virtual machine monitor". Proceedings of the 9th USENIX Security Symposium, Denver, CO, USA, p.129–144.

Rose, R. (2004)."Survey of System Virtualization Techniques". Disponível em http://www.robertwrose.com/vita/rose-virtualization.pdf. Acessado em 11/11/2006.

Rosenblum, M. e T. Garfinkel (2005). "Virtual machine monitors: current technology and future trends". Computer, v.38, n.5, p.39.

Smith, J. E. e R. Nair (2005). "Virtual Machines: Versatile Platforms for Systems and Processes": Morgan Kaufmann.

Sugerman, J., G. Venkitachalam, et al. (2001). "Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor". USENIX 2001. Boston, Massachusetts, USA. June 25–30. p. 1-14.

Vmware (2006)."VMware Server". Disponível em http://www.vmware.com/products/server/. Acessado em 15 Out. 2006.

Whitaker, A., M. Shaw, et al. (2002). "Scale and Performance in the Denali Isolation Kernel". Fifth Symposium on Operating System Design and Implementation. p. 195-209.

Xen (2006)."XEN Source". Disponível em http://www.xensource.com/. Acessado em 10 Out. 2006.