Análise de Desempenho do KVM Aplicado a Paravirtualização e Virtualização Assistida por Hardware Embasada pela RFC 2544

Everson Luis R. Lucion[1], Giulliano G. Minuzzi[1], Mauricio V. Almeida[1]

[1]Centro de Processamento de Dados (CPD)
Universidade Federal de Santa Maria (UFSM) – Santa Maria, RS – Brasil

{everson,gminuzzi,mauricio}@cpd.ufsm.br

Abstract. *Virtualization is essential for consolidating datacenters, NFV and cloud computing. In this work, experiments with KVM and Intel VT technology resources were performed to identify the best virtualization technique regarding paravirtualization and hardware-assisted virtualization based on RFC 2544. Throughput, loss and latency metrics were evaluated. The results show that the gains apply only to the features of the Processor. Network throughput in emulated devices does not show significant performance gain due to limitation of vCPU usage by Qemu and poor performance in binary translation. Paravirtualization has better performance, but the virtualized machine needs to be modified to be compatible, which can lead to compatibility and portability issues.*

Resumo. *A virtualização é essencial para consolidação datacenters, NFV e computação em nuvem. Neste trabalho foram realizados experimentos com o KVM e recursos da tecnologia Intel VT para identificar a melhor técnica de virtualização no que tange a paravirtualização e virtualização assistida por hardware, embasados pela RFC 2544. As métricas de throughput, perdas e latência foram avaliadas. Os resultados mostram que os ganhos se aplicam somente aos recursos do Processador. O throughput de rede em dispositivos emulados não apresenta ganho de desempenho significativo devido a limitação de uso de vCPU pelo Qemu e desempenho fraco na tradução binária. A paravirtualização apresenta melhor desempenho, porém a máquina virtualizada precisa ser modificada para ser compatível, podendo ocasionar problemas de compatibilidade e portabilidade.*

1. Introdução

A virtualização é a tecnologia fundamental em centros de dados e computação em nuvem. Empresas e instituições acadêmicas buscam consolidar seus datacenters e reduzir custos. Soluções como NFV (Network Functions Virtualization) também chegaram para impulsionar a virtualização de serviços como NAT, *firewall*, IPS, DNS e *caching*.

O Virtual Machine Monitor (VMM) é o componente principal da virtualização, a sua eficácia afeta o desempenho de todo sistema diretamente, podendo executar na maioria das vezes a paravirtualização ou virtualização assistida por *hardware*. O *Kernel-based Virtual Machine* (KVM) é uma solução completa de virtualização em

código aberto que suporta extensões de virtualização (Intel VT ou AMD-V). O KVM foi integrado ao Kernel do Linux em 2007 [Carvalho and Bellezi 2014].

As equipes de TI sempre estão preocupadas com o viés do desempenho. A busca pela otimização e ganho é constante, suscitando dúvidas em relação a melhor abordagem de virtualização a ser utilizada.

Neste trabalho foram realizados experimentos com máquinas virtualizadas pelo KVM e recursos da tecnologia Intel VT visando identificar a melhor técnica de virtualização, obtendo assim melhor desempenho nas operações de *network* I/O em enlaces *Gigabit.* Os experimentos abordam as métricas de vazão, perdas e latência embasados pela RFC 2544 que recomenda tamanhos de *frames* variados (64, 128, 512, 1024, 1208 e 1518 bytes). Todos os tamanhos de *frames* são usados em uma rede e os resultados devem ser conhecidos. A comparação é realizada entre o *host* físico e duas máquinas virtualizadas, uma paravirtualizada (driver Virtio) e outra com virtualização assistida por hardware (driver emulado pelo Qemu). A máquina física foi incluída no experimento servindo com parâmetro de desempenho ideal.

Os resultados mostram que apesar da Tecnologia Intel VT ter sido desenvolvida para "otimizar" o trabalho do VMM, os ganhos de performance se aplicam somente ao uso dos recursos do Processador. Os recursos de *throughput* de rede em dispositivos emulados não apresentam ganhos de performance significativos devido a limitação de uso de vCPU pelo Qemu e fraco desempenho da tradução binária. A paravirtualização apresenta melhor desempenho, porém a máquina virtualizada precisa ser modificada para ser compatível, podendo ocasionar problemas de compatibilidade e portabilidade.

Este trabalho está estruturado da seguinte forma: A Seção 2 discute os trabalhos já relacionados, a Seção 3 expõe o método, a técnica e as tecnologias que integram o experimento. A Seção 4 apresenta a metodologia proposta e a Seção 5 mostra os resultados dos experimentos. A Seção 6 conclui e discute trabalhos futuros.

2. Trabalhos Relacionados

Motika and Weiss (2012) analisaram através da ferramenta Iperf o desempenho do KVM com dispositivos de redes emulados e paravirtualizados. Relataram que a melhoria do *driver V*irtio na largura de banda de transmissão é de 54% e na largura de banda recebida é 58% em relação ao *driver* emulado. As máquinas virtuais apresentaram um melhor desempenho na recepção de dados e que a transmissão não teve desempenho satisfatório. A utilização da CPU foi alta em ambos ambientes, mais de 90% de utilização de CPU no *driver* paravirtualizado e um pouco maior no *driver* emulado. Não informaram largura de banda e não fizeram uso da RFC 2544.

Mais próximo a este trabalho está a proposta de Kolling et al. (2008), o qual teve como principal objetivo implementar os experimentos da RFC 2544 em sistema embarcado, que é um conjunto computacional dedicado e usa seu poder de processamento exclusivo para a função a qual foi projetado. Comparou o desempenho em testadores implementados em *software* com o protótipo desenvolvido em *hardware,* com o objetivo de validar a vazão máxima alcançada em cada um. Relatou que o experimento em *hardware* não alcançou a vazão máxima apenas para *frames* de 64 *bytes*, atingindo 71% do ideal. Em relação aos experimentos por *software*, todos os tamanhos de *frames* tiveram desempenho inferior, e que esta diferença se deve ao

elevado uso de CPU pelo aplicativo no processamento dos cabeçalhos dos *frames* Ethernet.

A nossa proposta apresenta em relação às anteriores, três inovações. Primeiro, analisa o desempenho e aspectos positivos e negativos de cada técnica de virtualização. Em segundo lugar identifica qual técnica apresenta melhor desempenho. Por último, relata porque a conversão binária fraca é o principal problema da virtualização.

3. A Virtualização por Método VMM

O KVM utiliza a tecnologia Intel VT em todas as técnicas de virtualização para aumento de performance nas operações ligadas ao processador, executando diretamente as operações das VMs no mesmo, evitando a tradução binária. Por outro lado o gerenciamento de dispositivos é efetuado pelo Qemu quando utilizada a virtualização total, já na paravirtualização os mesmos são gerenciados pelos Virtio *drivers*. O VMM é também conhecido como *hypervisor* e neste experimento muitas vezes é referenciado como máquina física.

3.1. Tecnologia de Virtualização Intel VT

Desenvolvida para oferecer melhor performance, a virtualização assistida por *hardware (HVM)* requer extensões de *hardware* Intel VT ou AMD-V, possibilitando que o VMM execute de forma isolada máquinas virtuais (VMs) com qualquer sistema operacional, necessitando que o processador ofereça suporte a tecnologia [Patel and Chaudhary 2014]. De acordo com Motika and Weiss (2012), a introdução desta tecnologia nos processadores visa reduzir o *overhead* gerado pela troca de contexto entre o VMM e o sistema hospedeiro, através da utilização de uma estrutura de dados que retém informações sobre o estado da CPU (*Virtual-Machine Control Structure* - VMCS).

A virtualização assistida por *hardware* substitui a tradução binária por dois modos de operação no processador, sendo estes o *root* que opera de forma semelhante ao processador habitual, responsável pela execução do VMM, e o *non-root* responsável somente pela execução de atividades das VMs, os quais, também, oferecem suporte aos quatro *rings* da arquitetura x86-64, porém o modo *non-root* dispõe de menos privilégios em qualquer *level* dos *rings* [Grinberg and Weiss 2012].

3.2. Técnicas de Virtualização

A Paravirtualização é uma técnica que consiste em modificar o núcleo do sistema convidado (*guest*) substituindo as chamadas privilegiadas do SO por chamadas ao *hypervisor* (*hypercalls*) [Shirinbab, Lundberg e Ilie 2014]. Dessa forma, o *guest* tem acesso direto ao *hardware* através da utilização de *drivers* que se comunicam com o *back end driver* do VMM. Na paravitualização, a VM precisa ser modificada para ser compatível, podendo ocasionar problemas de compatibilidade e portabilidade. No KVM, a paravirtualizacão é baseada na utilização dos Virtio *drivers,* separando o *driver* do *guest* da implementação por parte do *hypervisor* [Zhang et al. 2010].

Virtualização completa, também conhecida como emulada, executa um g*uest* sem necessidade de modificações de seu *kernel*, recebendo dispositivos emulados pelo VMM, o que garante a portabilidade e compatibilidade com outros sistemas.

A Intel agregou a virtualização assistida por *hardware* a seus processadores através das tecnologias Intel VT, para que o *hypervisor* possa mais eficientemente delegar o acesso a recursos restritos. A Intel VT otimiza a execução de operações gerenciadas pelo processador, mas seus *drivers de* rede (placas *ethernet*) e bloco (*drivers* de disco), entre outros, ainda são emulados pelo VMM e gerenciados pelo Qemu. *Drivers* como o e1000 são atribuído ao *guest* pelo VMM neste experimento. Existem outras opções como vmxnet3 e rtl8139.

4. Metodologia da Avaliação de Desempenho

O protocolo Ethernet IEEE 802.3 é usado em redes de computadores e telecomunicações, define a camada física e enlace. Neste sentido, conexões *ethernet* podem ser testadas para garantir o nível de desempenho necessário. A avaliação de desempenho e a forma dos resultados, podem ser embasados em metodologias definidas pela IETF e publicadas através da RFC 2544 (COSTA, 2008).

A RFC 2544 é um padrão internacional formulado pela RFC (Request For Comments) para avaliar o desempenho de dispositivos de redes. A RFC 2544 orienta que utilizem-se *frames* nos tamanhos 64, 128, 256, 512, 1024, 1280 e 1518 *bytes*, enviados durante um intervalo de tempo e por um número definido de repetições. Os *frames* menores têm uma vazão efetiva menor do que o dos *frames* maiores, devido à inclusão dos *bytes* de cabeçalho e espaço entre *frames*, não sendo contabilizados como dados.

Para implementar o conjunto dos experimentos, a RFC 2544 define um ***tester*** (dispositivo testador) um **DUT** (dispositivo sob teste) do inglês "*device under test*", conforme pode ser visto na Figura 1.

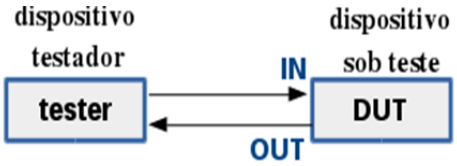

Figura 1. *tester e* DUT

4.1. Métricas de Avaliação de Desempenho

Muitas vezes pode ser essencial a realização de experimentos de desempenho, onde é necessário analisarmos determinados cenários de operações e observarmos os resultados obtidos. As métricas de desempenho avaliadas neste experimento são as seguintes:

- *Throughput* ou vazão: Taxa mais rápida na qual a contagem de *frames* reflete o máximo de tráfego de dados que podem ser manipulados sem perdas;
- Latência: A medida do atraso da extremidade de uma rede (*link*) ou dispositivo e vice-versa. O atraso é a soma de processamento nos elementos de rede e dos atrasos de propagação ao longo do meio de transmissão;
- Taxa de perda: A taxa de perda refere-se à porcentagem de *frames* que nunca foram recebidos em relação aos *frames* que foram transmitidos com sucesso a partir da fonte;
- Desempenho da CPU: Foram recolhidos dados sobre o desempenho das CPUs

durante os experimentos de t*hroughput,* nos eventos de transmissão e recepção do DUT.

4.2. Descrição dos Experimentos

Neste trabalho, a aferição do *throughput* ou vazão usou a ferramenta de *benchmarking* Nuttcp e o protocolo UDP com tamanhos de *frames* definidos pela RFC 2544 em cada direção do DUT (transmissão e recepção). Foram efetuados 10 experimentos consecutivos com 60 segundos calculando-se a sua média em Mbps.

Nos experimentos de latência, utilizou-se a ferramenta "ping" para efetuar 10 sequencias com envio de 100 *frames* conforme a RFC 2544. Para medir a latência, os experimentos foram feitos através de um *loop* e apresentados em milisegundos.

Na taxa de perda, o DUT foi aferido conforme a RFC 2544. Neste experimento também foi utilizada a ferramenta Nuttcp. Foram efetuados 10 eventos consecutivos com 60 segundos cada e a média foi apresentada em porcentagem.

O Nuttcp é uma ferramenta de medição de desempenho de rede. O uso mais comum é determinar o *throughput* da camada de rede TCP (ou UDP). Ele fornece informações adicionais, como a utilização da CPU do transmissor e do receptor

Os experimentos foram realizados em três cenários. O primeiro é o ***tester***: SGI 1U, RAM de 13 GB, 8 @ 2.00GHz. O segundo é o **VMM:** DELL PowerEdge R620, RAM 129 GB, 32 @ 2.60GHz. Em terceiro, as duas **VMs**, com 4 processadores, memória RAM de 8 GB cada e com interfaces Gigabit Ethernet. O KVM-QEMU versão 2.1.2, Nuttcp versão 6.1.2. e o SO Debian Jessie 3.16.0-4-amd64.

5. Resultados e Discussão

Na Figura 2(a), temos os resultados relacionados a taxa de vazão (DUT IN). Nota-se que a vazão da máquina física ou (VMM) e a paravirtualizada (PV) se mantiveram relativamente iguais. A virtualização assistida por *hardware* (HVM) apresentou um desempenho excelente nos *frames* de 1280 bytes (949.11 Mbps) e 1518 *bytes* (952.33 Mbps), e satisfatório nos demais tamanhos.

O desempenho da HVM não foi considerado satisfatório no (DUT OUT) em todos tamanhos de *frames* testados. Este comportamento pode ser explicado por conta do *driver* emulado, em que todo o I/O é manipulado pelo Qemu, pela limitação de uso de vCPU e fraco desempenho na tradução binária. A HVM apresentou uma vazão máxima de 189.94 Mbps com *frames* de 1518 *bytes*. A VMM e a PV apresentaram excelentes resultados. A Figura 2(b) demonstra estes resultados.

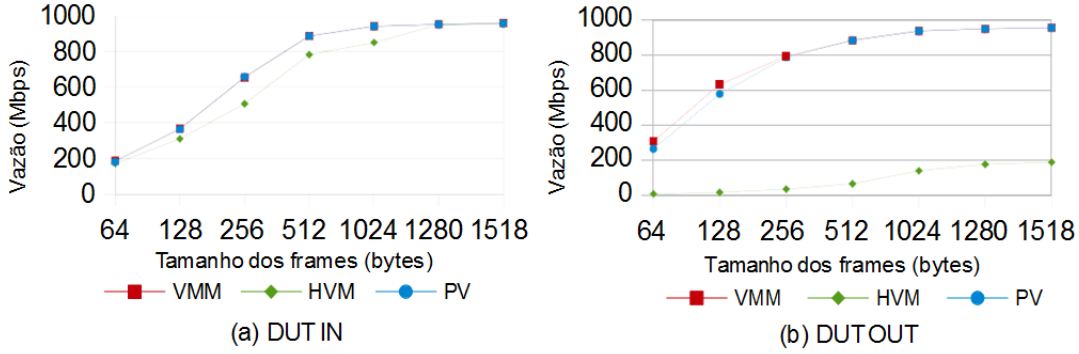

Figura 2. Taxa de vazão ou throughput

Quanto ao uso da CPU na recepção de dados (DUT IN), ambas tiveram um resultado satisfatório, onde o uso de CPU se manteve abaixo de 70% de processamento durante os testes de vazão. A Figura3(a) mostra os resultados.

Já no envio de dados (DUT OUT), observado na Figura 3(b), a VMM e a PV tiveram resultados semelhantes. A PV teve um pico máximo de 97.04% de uso nos *frames* de 64 *bytes*. A HVM não teve resultado satisfatório, pois o Qemu não foi projetado para ser *multithreading* na tradução binária, acarretando alto consumo de processamento (99%) de uma única CPU e, por consequência, baixa vazão de dados.

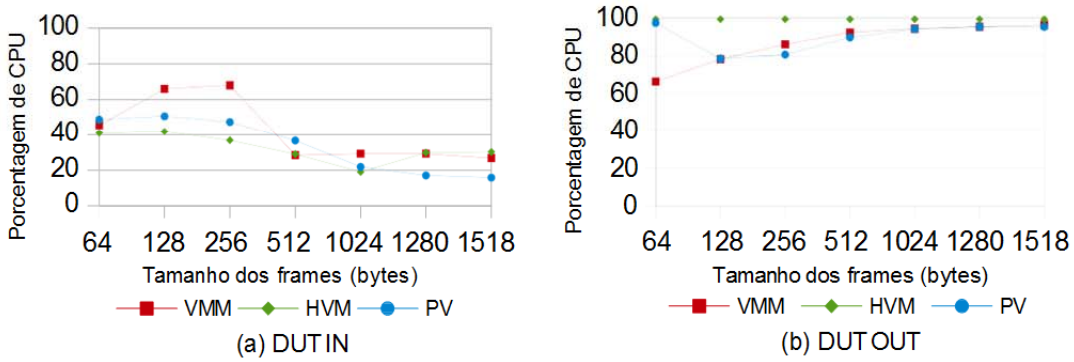

Figura 3. Desempenho da CPU

Na Figura 4(a) observa-se a taxa de perda de *frames* durante o recebimento de dados (DUT IN) nos experimentos de vazão. O HVM apresentou alta perda nos *frames* de 64, 128, 256, 512 e 1024 bytes, com pico de 23,33 % de erros em *frames* de 256 bytes, porém apresentou bons resultados nos frames de 1280 e 1518 bytes, com 0.1966% e 0,55% de perdas respectivamente. A paravirtualizada conseguiu atingir bons níveis de vazão com baixa perda de *frames* (pico de 2,5% de perda com *frames* de 64 *bytes*). A perda de *frames* na máquina física pode ser considerada desprezível por conta da alta vazão alcançada.

A Figura 4(b) representa a perda de *frames* durante experimentos (DUT OUT). Demonstra que a PV apresenta um pico na taxa de erros em *frames* de 128 bytes, com 3,37% de perda. A HVM não apresentou perda nos experimentos OUT devido a baixa vazão de dados ocasionado pela limitação de uso de uma única virtual CPU nas operações de network I/O. A taxa de perda da VMM teve pico de 0.378% com *frames* de 128 *bytes,* sendo considerado muito baixo.

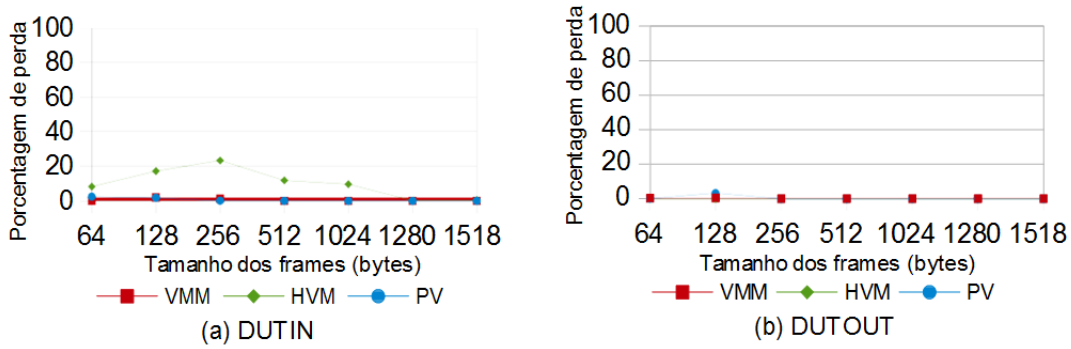

Figura 4. Perda de *frames*

Os experimentos de latência indicam que a camada de virtualização traz um acréscimo no tempo de resposta de processamento dos *frames* pelas VMs. Isso fica evidente na HVM, em que a latência é quase o dobro do VMM em alguns casos, novamente por conta do *driver* de rede emulado. A Figura 5 apresenta os resultados.

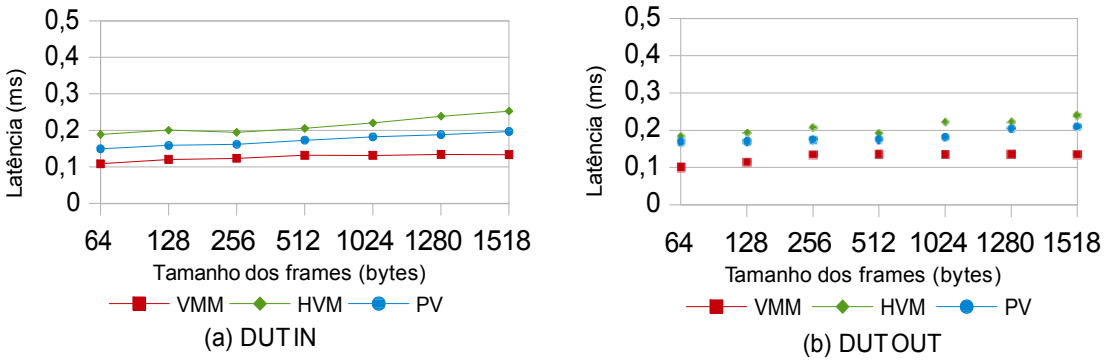

Figura 5. Latência

O principal problema da virtualização total é seu desempenho fraco na conversão binária. Acelerar a conversão binária é difícil. Segundo [Kiyanovski 2016] há quatro razões para o *driver* emulado e1000 (usado neste experimento) ser mais lento que o *driver* *V*irtio: 1. Ele quebra o segmento TCP/UDP para o tamanho dos *frames* MTU; 2. Ele calcula o TCP/UDP *checksum* para cada segmento; 3. Copia os dados do *guest* para o *host* antes de enviá-los; 4. Ele lida com interrupções de coalescência (combinação de partições adjacentes de memória livre) deficientemente para cenários de *throughput*.

Na paravirtualização, o *driver* de rede Virtio deixa a quebra do segmento e cálculo do *checksum* para o *host* físico, o qual executa ambas as tarefas muito mais rapidamente. O Virtio *driver* não executa a cópia dos dados, uma vez que os *buffers* não serão alterados até eles serem manipulados pelo dispositivo. E finalmente, cenários de *throughput* provocam interrupções a uma taxa consideravelmente inferior a emulação e1000, reduzindo a sobrecarga causada por interrupções.

6. Conclusão e Trabalhos Futuros

Visando reduzir custos e a complexidade dos ambientes de TI, a virtualização é uma tecnologia essencial para consolidação de serviços de redes, *datacenters,* NFV e computação em nuvem. Este experimento mostra a comparação de desempenho de rede em *link* Gigabit do VMM e máquinas virtuais com KVM, ambas com recursos da tecnologia Intel VT. As métricas avaliadas foram: *throughput*, perda de *frames,* latência e utilização de CPU. A RFC 2544 foi usada para validar as características de desempenho.

Os resultados mostram que a paravirtualização apresenta melhor desempenho e possui resultados muito similares ao *host* físico nas taxas de vazão recebidas e enviadas, perda de *frames* e latência. A virtualização assistida por *hardware* apresentou um bom desempenho nas taxas de vazão recebidas e não satisfatório nas taxas envidadas.

A Tecnologia Intel VT foi desenvolvida para otimizar o desempenho das máquinas virtuais, mas os ganhos se aplicam somente ao uso dos recursos do

Processador. O *throughput* de rede em dispositivos emulados não apresenta ganho de desempenho significativo, devido a limitação de uso de vCPU pelo Qemu e seu desempenho fraco na tradução binária. As principais razões pela lentidão na tradução binária são a quebra do segmento TCP/UDP em *frames* e o cálculo de *checksum* ao envidar dados. A paravirtualização apresenta melhor desempenho, porém a máquina virtualizada precisa ser modificada para ser compatível, podendo ocasionar problemas de compatibilidade e portabilidade.

Para qualquer sistema operacional moderno, os *drivers* gerenciados pelo Qemu só deveriam ser usados provisoriamente como um suporte para colocar o sistema em funcionamento até que se possa instalar e mudar para os Virtio *drivers,* em casos que não possam ser instalados juntamente com o Sistema Operacional. Observa-se que otimizar *drivers* emulados para executar em múltiplas vCPU não é prioridade dos desenvolvedores do Qemu, por isso recomenda-se mudar para os Virtio *drivers* caso não haja problemas de compatibilidade e portabilidade.

Trabalhos futuros incluem estudos para diminuir a diferença entre a paravirtualização e a virtualização total, abordando o principal problema da virtualização encontrada neste experimento, a conversão binária fraca.

Referências

Carvalho, F. L. and Bellezi, M. A. (2014). Avaliação de Desempenho dos Hypervisors XEN e KVM utilizando Administração Simplificada através do LibVirt. T.I.S. São Carlos, v. 3, n. 1, p. 88-101.

Costa, G. H. Da. (2008). Métricas para Avaliação de Desempenho em Redes QoS sobre IP. Curso De Especialização Em Tecnologias, Gerência E Segurança De Redes De Computadores. UFRGS.

Grinberg, S. and Weiss, S. (2012). Architectural virtualization extensions: A systems perspective. Computer Science Review. November 2012, Vol.6(5-6), pp.209-224.

Kiyanovski, A. (2016). Student research poster: Network controller emulation on a sidecore for unmodified virtual machines. Parallel Architecture and Compilation Techniques (PACT), 2016 International Conference on.

Kolling, M. L. et al. (2008). Verificação em Hardware de Componentes de Comunicação. IX Simpósio em Sistemas Computacionais. Departamento de Informática. Universidade de Santa Cruz do Sul (UNISC). p. 143-150.

Motika G. and Weiss S. (2012). Virtio Network paravirtualization driver: Implementation and performance of a de-facto standard. Computer standards &

interfaces, vol. 34, pg. 36-47. Elsevier.

Patel, M. and Chaudhary, S. (2014) Survey on a combined approach using prediction and compression to improve pre-copy for efficient live memory migration on Xen. Parallel, Distributed and Grid Computing (PDGC), 2014 International Conference on, pages 445-450. IEEE.

Shirinbab, S., Lundberg, L. and Ilie, D. (2014) Performance Comparison of KVM, VMware and XenServer using a Large Telecommunication Application. The Fifth International Conference on Cloud Computing GRIDS and Virtualization.