IARA ROBENI DE OLIVEIRA SILVA

AVALIAÇÃO DE HIPERVISORES QUANTO AO DESEMPENHO DE REDE

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Orientador:
Prof. Joaquim Quinteiro Uchôa

LAVRAS - MG

2010

IARA ROBENI DE OLIVEIRA SILVA

AVALIAÇÃO DE HIPERVISORES QUANTO AO DESEMPENHO DE REDE

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

APROVADA em 07 de junho de 2010.

_____
Prof. Bruno de Oliveira Schneider UFLA

_____
Profª. Marluce Rodrigues Pereira UFLA

_____
Prof. Joaquim Quinteiro Uchôa
(Orientador)

LAVRAS - MG

2010

Dedico este trabalho ao meu “grande” pai,
que com esforço e dedicação
me possibilitou um estudo de qualidade.

AGRADECIMENTOS

Em primeiro lugar a Deus, por ter me dado forças e proteção durante toda graduação.

Ao meu pai, por tornar tudo isso possível!

A Zezé e Vó Titida, por estarem sempre ao meu lado em toda vida estudantil.

A toda a minha família, mãe, pai, irmã, tios, tias, primos e primas, são todos mais que queridos!

Ao meu orientador Prof. Joaquim Quinteiro Uchôa, pela oportunidade e principalmente pela paciência durante o desenvolvimento do trabalho.

Ao Alkmim, pela grande paciência em me ensinar Linux.

Aos companheiros do Grupo de Capoeira "I Lá Vou Eu", pelo apoio e paciência, nos momentos de desespero, amigos mais que especiais.

Aos professores e funcionários do DCC – Departamento de Ciência da Computação.

A todos os amigos, de longe e de perto. Sei que posso sempre contar com vocês.

Enfim, a todos que estiveram ao meu lado durante esta etapa da minha vida. Muito Obrigada!

SUMÁRIO

LISTA DE FIGURAS

RESUMO

AVALIAÇÃO DE HIPERVISORES QUANTO AO DESEMPENHO DE REDE

Com o desenvolvimento computacional dos últimos anos o uso da virtualização cresceu de forma extraordinária. Inúmeros métodos e ferramentas tem sido propostos e utilizados, fazendo com que a escolha de um deles não seja trivial. Dentre os fatores a considerar, o desempenho de rede é fundamental, já que os computadores dependem fortemente de comunicação entre si. Este trabalho apresenta a avaliação dos hipervisores Xen, VirtualBox e QEMU quanto ao desempenho de rede. Para isso foi utilizado o *benchmark Iperf*, a fim de verificar a taxa de transmissão máxima alcançável por cada máquina virtual. Foram realizados dez testes para cada máquina virtual criada com os hipervisores em questão. Através de análises dos resultados obtidos, pôde-se observar um grande desempenho alcançado pelo VirtualBox, equivalentes ao do Xen, face ao seu grande desenvolvimento nos últimos anos.

Palavras-chaves: Virtualização. Hipervisores. Desempenho de rede.

1 INTRODUÇÃO

Neste capítulo será descrita uma breve introdução do trabalho, as motivações que levaram à realização do mesmo e os objetivos a serem alcançados ao término dos estudos.

1.1 Contextualização e Motivação

Atualmente, a virtualização em ambientes computacionais constitui um tema de pesquisas e desenvolvimento em áreas da computação. A virtualização consiste na técnica de, em uma mesma máquina física, criar máquinas virtuais, com sistemas operacionais diferentes executando simultaneamente. Os recursos das máquinas físicas são compartilhados pelas máquinas virtuais, de modo que, a quantidade de máquinas virtuais criadas depende da quantidade de recursos do sistema físico e de quanto desses recursos serão utilizados pelas mesmas.

A virtualização teve sua origem em meados das décadas de 60 e 70, quando foi estabelecido um novo conceito de utilização de máquinas de grande porte, definindo uma técnica que possibilitava a divisão de um mesmo *hardware* por diversas máquinas virtuais (SANTOS, 2008). Com o passar do tempo, a evolução de recursos (poder computacional, memória, armazenamento, entre outros) permitiu que servidores de pequeno porte e computadores pessoais tivessem condições de aplicar a virtualização, colaborando com o aperfeiçoamento da técnica. A partir de então, o uso da virtualização se deu de forma crescente (SOUZA, 2006).

No contexto atual, o interesse em virtualizar sistemas não está apenas no fato de permitir o uso de um mesmo sistema por vários usuários concomitantemente, mas os principais interesses estão na segurança, confiabilidade e disponibilidade, custo, adaptabilidade, balanceamento de carga

e suporte a aplicações legadas (programas antigos e de difícil atualização, sem versões mais recentes).

Dentre as vantagens oferecidas pela utilização da virtualização podem-se destacar:

Segurança: as máquinas virtuais são isoladas e independentes umas das outras, podendo ser utilizadas como ambientes de testes.

Redução de custos: necessita-se de menos *hardware* para gerenciar uma rede.

Melhor aproveitamento de *hardware*: as máquinas virtuais compartilham o *hardware* da máquina física.

Facilidade de migrar ambientes: evitando a reinstalação e reconfiguração dos sistemas a serem migrados.

Possibilidade de simular redes de computadores com menor demanda de *hardware*: é possível a simulação de redes virtuais utilizando o mesmo *hardware*.

Para criação de máquinas virtuais é introduzida uma camada de *software* entre o *hardware* e o sistema operacional, o monitor de máquinas virtuais (MMV) ou hipervisor. Os hipervisores são os responsáveis por criar e gerenciar as máquinas virtuais, controlando o acesso dos sistemas operacionais visitantes aos dispositivos de *hardware*. Diversos métodos e ferramentas de virtualização tem sido propostos, o que torna difícil a escolha de uma solução que atenda a todos os requisitos de cada ambiente. Dentre os fatores a considerar, o desempenho de rede é fundamental, já que os sistemas atuais dependem fortemente de comunicação entre si.

Dessa maneira, torna-se extremamente importante o estudo e análise de ferramentas de virtualização, visando a escolha adequada da tecnologia a ser utilizada de acordo com a aplicação em questão.

1.2 Objetivos

Este trabalho tem como objetivo geral analisar o desempenho de rede de algumas ferramentas de virtualização. Dentre as ferramentas disponíveis atualmente foram escolhidas aquelas que estivessem em crescente utilização no mercado e que constituíssem soluções livres de virtualização. Dentre os objetivos específicos estão:

- ✓ Analisar o desempenho de rede dos hipervisores Xen, VirtualBox e QEMU.
- ✓ Medir o tráfego de rede entre as máquinas analisadas por meio do *benchmark Iperf*.
- ✓ Depois de realizados os testes comparar e verificar qual melhor ferramenta de virtualização em relação ao aspecto analisado.
- ✓ Relacionar os resultados encontrados com as diferentes técnicas de virtualização adotadas.

1.3 Estrutura do Trabalho

O presente trabalho divide-se em seis capítulos, os quais visam a abordagem de questões relacionadas à virtualização, hipervisores e desempenho de rede, desta forma, os capítulos estão estruturados da seguinte maneira: o Capítulo 2 apresenta os conceitos gerais sobre virtualização, definições de monitores de máquinas virtuais e as diferentes formas de virtualização. O Capítulo 3 apresenta alguns hipervisores livres destacando suas principais características, dentre eles Xen, KVM, QEMU e VirtualBox, apresenta ainda, trabalhos relacionados com avaliação de ferramentas de virtualização. O Capítulo 4 relata a metodologia utilizada para o desenvolvimento do trabalho. O Capítulo 5 descreve os resultados e discussões obtidas com este estudo. Por fim,

o Capítulo 6 apresenta as conclusões do trabalho desenvolvido, suas contribuições e trabalhos futuros.

2 VIRTUALIZAÇÃO

Neste capítulo serão apresentadas algumas definições relacionadas à virtualização, bem como suas principais características e funcionalidades.

2.1 Conceitos Gerais

A virtualização pode ser definida como a criação de um ambiente virtual que simula um ambiente real, ou seja, é a técnica que permite que em uma mesma máquina física sejam executadas diferentes máquinas virtuais (MV). Os recursos das máquinas físicas são compartilhados pelas máquinas virtuais, são eles que definem quantas MV o sistema pode suportar.

Para criação de máquinas virtuais é introduzida uma camada de *software* sobre a máquina física, denominada monitor de máquinas virtuais (MMV). Os MMVs ou hipervisores são responsáveis pela criação e gerenciamento das máquinas virtuais, além de controlar o acesso dos sistemas operacionais visitantes aos dispositivos de *hardware*. Eles são capazes de criar máquinas virtuais independentes, cada uma funcionando como se fosse um computador real. A estrutura de um sistema virtualizado é ilustrada na Figura 2.1, que mostra o MMV intermediando o acesso dos sistemas operacionais virtualizados ao *hardware*.

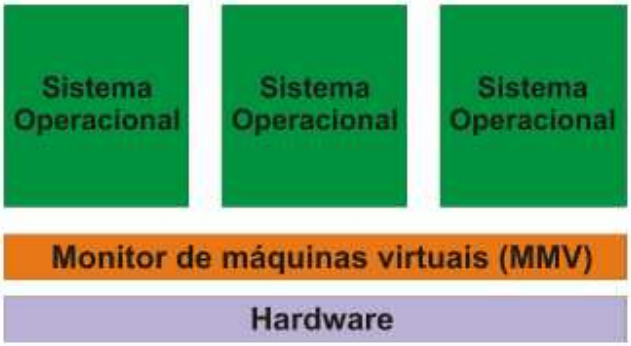

Figura 2.1: Estrutura de um sistema virtualizado.
Fonte Schimidt (2007)

De uma maneira mais formal, uma máquina virtual é definida como sendo uma duplicata eficiente e isolada de uma máquina real (POPEK E GOLDBERG, 1974). Uma duplicata é uma cópia idêntica de uma máquina original, ou seja, a máquina virtual deve ter o mesmo funcionamento de uma máquina real. Segundo Popek e Goldberg(1974), o termo isolamento significa que a máquina virtual trabalha como se fosse um computador independente. Com isso, o usuário ao utilizar uma máquina virtual não encontrará diferença em relação a uma máquina real. Além disso, possíveis falhas de máquinas virtuais não serão propagadas para as outras e não irão interferir no sistema hospedeiro (WLODARZ, 2007).

Um ambiente de máquina virtual consiste de três partes básicas (LAUREANO, 2008):

- o sistema real, nativo ou hospedeiro (*host system*), que contém os recursos reais de *hardware* e *software* do sistema;
- o sistema virtual, também denominado sistema convidado (*guest system*), que executa sobre o sistema virtualizado; em alguns casos, vários sistemas virtuais podem coexistir, executando simultaneamente sobre o mesmo sistema real;

- a camada de virtualização, hipervisor, ou monitor (MMV – *Virtual Machine Monitor*), que constrói as interfaces virtuais a partir da interface real.

2.2 Monitor de Máquinas Virtuais (MMV)

O monitor de máquinas virtuais ou hipervisor é a camada de *software* que faz a interface entre os recursos da máquina real e a máquina virtual. O MMV é responsável por criar a máquina virtual e gerenciar seu funcionamento. Ele é capaz de criar várias máquinas virtuais simultaneamente, cada uma com seu sistema operacional, funcionando independente uma da outra. O MMV também fornece proteção para o sistema hospedeiro impedindo que a máquina virtual realize alguma operação indevida no *hardware*. As principais funções do monitor de máquinas virtuais são (SILVA, 2007):

- definir o ambiente de máquinas virtuais;
- alterar o modo de execução do sistema operacional convidado de privilegiado para não privilegiado, e vice-versa;
- emular as instruções e escalonar o uso da CPU para as máquinas virtuais;
- gerenciar acesso aos blocos de memória e disco destinados ao funcionamento das máquinas virtuais;
- intermediar as chamadas de sistema e controlar acesso a outros dispositivos como CD-ROM, drives de disquete, dispositivos de rede, dispositivos USB.

Também é função do monitor de máquinas virtuais escalonar qual máquina virtual vai executar a cada momento, de forma semelhante a um escalonador de processos do sistema operacional.

Para funcionar de forma eficiente, um hipervisor deve atender a alguns requisitos básicos: deve prover um ambiente de execução aos programas idêntico ao da máquina real, deve ter controle sobre os recursos do sistema real. Segundo Popek e Goldberg (1974), os hipervisores devem possuir três características fundamentais:

Eficiência: é extremamente importante que um grande número de instruções do processador virtual seja executada diretamente pelo processador real, sem que haja intervenção do monitor. As instruções que não puderem ser tratadas pelo processador real precisam ser tratadas pelo monitor.

Integridade: todas as requisições aos recursos de *hardware* devem ser alocadas explicitamente pelo monitor (memória, processamento, etc).

Equivalência: o monitor deve prover um comportamento de execução semelhante ao da máquina real para o qual ele oferece suporte de virtualização, salvo haja a necessidade de se fazer alterações na disponibilidade de recursos da máquina.

Com o objetivo de maximizar o desempenho, o monitor de máquinas virtuais pode, sempre que possível, permitir que o sistema convidado execute as instruções diretamente sobre o *hardware* da máquina física em modo usuário (LAUREANO, 2004). Caso não seja possível, o MMV toma o controle do sistema, realiza as interpretações necessárias, executa a instrução sobre o *hardware* e devolve o controle para a máquina virtual.

O monitor de máquinas virtuais pode ser implementado de duas formas. Pode ser executado diretamente sobre o *hardware* da máquina real ou pode ser executado sobre o sistema operacional da máquina hospedeira (LAUREANO, 2003).

O primeiro caso, chamado de monitor nativo ou de tipo 1 (Figura 2.2), o monitor de máquinas virtuais é implementado entre o *hardware* e os sistemas convidados. Nesse caso, todos os recursos são gerenciados pelo MMV, cada

máquina virtual se comporta como uma máquina física completa que pode executar o seu próprio sistema operacional (LAUREANO, 2004). Como resultado é obtido um conjunto de computadores virtuais executando sobre um mesmo sistema físico. Se houver a necessidade de comunicação entre os sistemas, essa deverá ser feita via rede. Alguns exemplos de sistemas que empregam esta abordagem são *VMware ESX Server* (VMWARE, 2010) e o ambiente Xen (XEN, 2010).

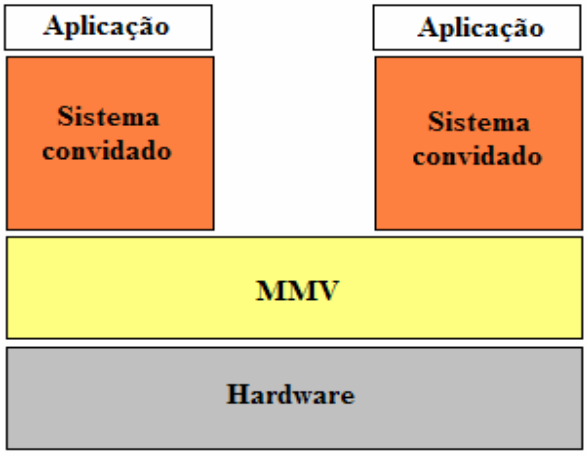

Figura 2.2: Monitor de máquinas virtuais de tipo 1.
Adaptado de Laureano, 2003.

O segundo caso, chamado de monitor convidado ou de tipo 2 (Figura 2.3), o monitor de máquinas virtuais é um processo sendo executado no sistema operacional hospedeiro, denominado sistema anfitrião. Este sistema é apenas responsável por gerenciar as máquinas virtuais. Várias outras aplicações podem ser executadas no sistema anfitrião junto com o monitor de máquinas virtuais (LAUREANO, 2004). Neste caso os recursos do *hardware* são divididos entre as máquinas virtuais e entre os outros serviços que estão sendo executados no sistema anfitrião. Exemplos de sistemas que adotam esta estrutura incluem o

VMware Workstation (VMWARE, 2010), o QEMU (QEMU, 2010) e o VirtualBox (VIRTUALBOX, 2010).

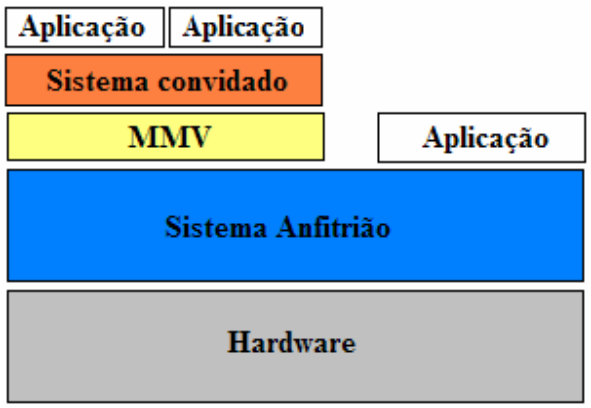

Figura 2.3: Monitor de máquinas virtuais de tipo 2.
Adaptado de Laureano, 2003.

Existem ainda as abordagens híbridas que são o tipo 1 e tipo 2, porém com algumas otimizações a fim de melhorar o desempenho das máquinas virtuais (LAUREANO, 2004). As otimizações inseridas em sistemas nativos ou de (tipo 1) permitem que as máquinas virtuais tenham acesso direto ao *hardware*, para isso o núcleo do sistema convidado e o MMV devem ser modificados. Já em sistemas convidados (tipo 2), podem ser inseridos três tipos de modificações: o sistema convidado pode ter acesso direto ao sistema anfitrião, o sistema convidado pode ter acesso direto ao *hardware* ou o MMV pode acessar diretamente o *hardware*.

A classificação apresentada nesta seção é realizada quanto à forma de implementação dos MMV. É possível também classificar quanto ao tipo de virtualização, este assunto será tratado na próxima seção.

2.3 Tipos de Virtualização

A virtualização pode ser classificada levando em consideração a modificação ou não dos sistemas operacionais convidados. Basicamente, os tipos de virtualização são: virtualização completa, paravirtualização, e emulação de *hardware*.

Na **virtualização completa ou total**, o sistema operacional hóspede pode ser executado sem nenhuma modificação, replicando virtualmente toda arquitetura do *hardware*. Neste tipo de virtualização, é criada uma cópia do *hardware* de modo que o sistema operacional convidado trabalhe como se estivesse sendo executado sobre o *hardware* real. Um ponto importante que deve ser considerado é que este tipo de virtualização necessita de um *hardware* com características específicas já que as instruções de execução privilegiada, como por exemplo, as de acesso a I/O, devem ser interceptadas e serem executadas de acordo com os critérios definidos pelo MMV. A vitualização completa só ocorre dentro da mesma arquitetura, dessa forma, uma máquina física x86 só poderá virtualizar a arquitetura x86.

A simulação completa do *hardware* feita por esta técnica de virtualização, geralmente, simula dispositivos padrões do mercado de modo a facilitar a instalação e configuração dos sistemas virtualizados. E esta simulação, se repete também para vídeo, *chipset* e discos rígidos. Isto quer dizer que independente dos modelos dos dispositivos da máquina física, os sistemas convidados sempre reconhecerão o dispositivo virtual como de um único modelo. São exemplos dessa tecnologia: Linux KVM (KVM, 2010), VirtualBox, VMware ESX, VMware Workstation. A Figura 2.4 apresenta a estrutura de um sistema que utiliza a virtualização total, onde pode ser observada uma camada de gerenciamento, que representa as aplicações responsáveis pela criação, pausa,

migração, destruição das máquinas virtuais, entre outras tarefas associadas à virtualização.

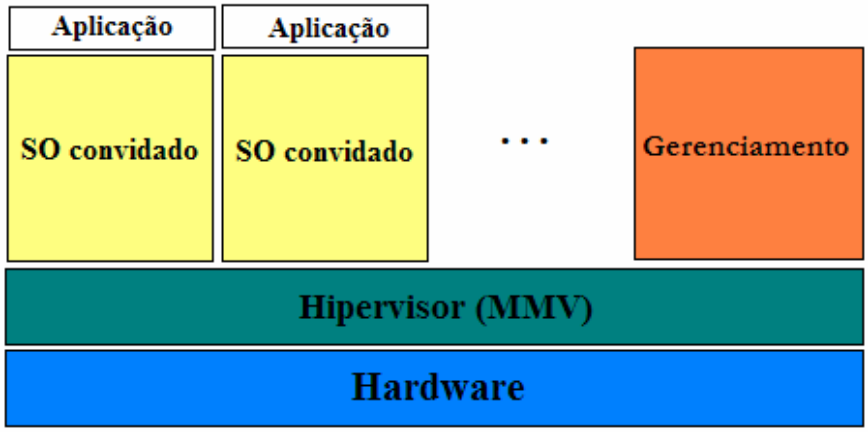

Figura 2.4: Estrutura de um sistema com virtualização total.
Adaptado de Jones, 2006.

Na **paravirtualização**, o *kernel* da máquina hóspede deve ser modificado para permitir que o monitor de máquinas virtuais execute tarefas protegidas. O *kernel* da máquina física também deve ser modificado devido às modificações feitas no *kernel* das máquinas hóspedes (URSCHEI, 2007). Estas modificações no *kernel* são necessárias para suportar uma virtualização eficiente (HUANG, 2007). Neste tipo de virtualização o MMV permite que as máquinas hóspedes se comuniquem diretamente com o *hardware*, sendo assim a perda de performance da paravitualização tende a zero. São exemplos desta tecnologia: Xen, VMware ESX e Lguest (LGUEST, 2010). A estrutura de um sistema paravirtualizado pode ser observada na Figura 2.5, onde existe uma camada fina, representada na cor laranja, que indica as ligações existentes entre os SO convidados e os hipervisor em questão.

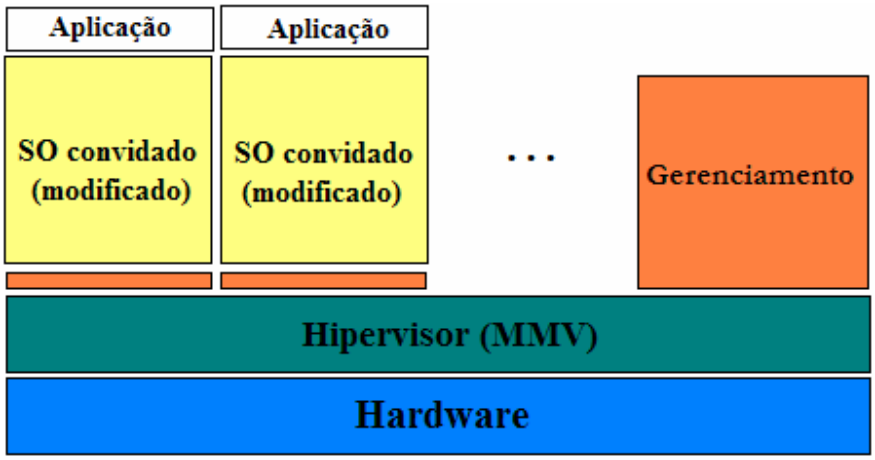

Figura 2.5: Estrutura de um sistema paravirtualizado.
Adaptado de Jones, 2006.

A **emulação de *hardware*** é considerado o tipo de virtualização mais complexo, devido ao fato de ter que emular de maneira precisa o comportamento de um *hardware*. Uma das vantagens desta técnica é a capacidade de emulação de qualquer arquitetura em uma outra. Com a emulação de *hardware* é possível rodar múltiplas máquinas virtuais, cada uma simulando um processador diferente (JONES, 2006). Entretanto, o problema deste tipo de virtualização é a lentidão com a qual a emulação acontece, pois as instruções devem ser modificadas para que possam ser aplicadas no *hardware* real. Segundo Jones (2006), o sistema emulado pode chegar a ser até 1000 vezes mais lento que o suposto *hardware* real. São exemplos desta tecnologia Bochs (BOCHS, 2010) e QEMU. A Figura 2.6 apresenta a estrutura de um sistema que utiliza emulação de *hardware*.

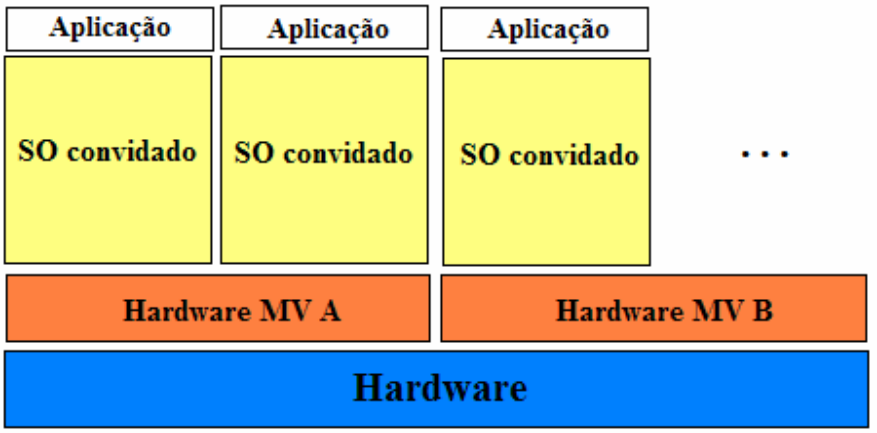

Figura 2.6: Estrutura de um sistema emulado.

Adaptado de Jones, 2006.

3 HIPERVISORES LIVRES

O objetivo deste capítulo é apresentar os hipervisores (MMV): Xen, KVM, QEMU e VirtualBox ressaltando suas características e principais configurações.

3.1 Xen

O Xen é um monitor de máquinas virtuais, para plataformas x86, desenvolvido pela Universidade de Cambridge, disponível desde 2003, e que possui código fonte aberto sob a licença GPL. Ele permite que múltiplos sistemas operacionais compartilhem recursos de *hardware* com garantia de desempenho e funcionalidades equivalentes à maquina real (BARHAM, 2003). O Xen utiliza a técnica da paravirtualização na criação de suas máquinas virtuais e a partir da versão 3.0 passou a oferecer também a virtualização total, desde que executado sobre um *hardware* HVM (*Hardware Virtual Machine*) , ou seja, os sistemas operacionais não precisam ser modificados, desde que os processadores apresentem arquitetura HVM (*hardware* com suporte à virtualização).

(SANTOS, 2008). O Xen apresenta-se como uma das soluções mais populares, já tendo sido alvo de avaliações de desempenho (URSCHEI, 2007).

Na paravirtualização, o sistema operacional que possui o monitor de máquinas virtuais é chamado de Dom0 (domínio 0) ou máquina hospedeira, as máquinas virtuais são chamadas de Dom1 (domínio 1) ou máquinas hóspedes. O *kernel* das máquinas hóspedes (domínio 1) são modificados de modo que o MMV execute as tarefas no modo protegido, ou seja, as máquinas hóspedes não acessam o *hardware* diretamente. O hipervisor Xen é configurado para executar no nível 0 (*ring 0*), somente ele tem acesso direto ao *hardware*, enquanto as máquinas hóspedes executam em nível 1 (*ring 1*), e não possuem acesso direto

ao *hardware*. Desse modo, o hipervisor funciona como um escalonador de acesso ao *hardware*. Assim como o *kernel* de cada máquina hóspede foi modificado, o *kernel* da máquina física (domínio 0) também necessita de um *kernel* modificado (URSCHEI, 2007).

Existem quatro fatores importantes na paravirtualização de arquiteturas x86: os gerenciamentos de memória, de CPU, dos dispositivos de E/S e de rede (BARHAM, 2003).

Em relação à memória, o Xen é o responsável por gerenciar a alocação de memória física para as máquinas virtuais e garantir que uma máquina virtual não acesse a região de memória de outra máquina. Em cada máquina virtual criada pelo Xen, o monitor aloca uma tabela de página de memória, a partir daí utiliza o mecanismo *hypercall* para fazer as atualizações. O monitor permite acesso direto das máquinas virtuais às páginas de memória, porém em modo de leitura. No gerenciamento de dispositivos de E/S o Xen emula o *hardware*, como é tipicamente feito em virtualização completa. O Xen provê uma abstração de um roteador virtual, onde cada domínio possui uma ou mais VIF (*Virtual Network Interface*) que seriam as placas de rede de cada sistema hospedado.

3.2 KVM

A abordagem utilizada pelo KVM (*Kernel – based Virtual Machine*) é a de tornar o *kernel* Linux um hipervisor e sua implementação é feita através de módulos carregáveis, visando maior simplicidade e desempenho (HABIB, 2008). Ao adicionar capacidade de virtualização a um padrão *kernel* Linux, o ambiente virtualizado pode se beneficiar de todos os trabalhos em curso sobre o *kernel* Linux.

O KVM é uma solução de virtualização completa, baseada em partes do QEMU. A diferença entre os dois está no fato de que o QEMU emula os

diferentes tipos de CPU e o KVM usa os recursos de virtualização disponíveis nos processadores e facilita a vida da máquina virtual.

Os processos Linux apresentam dois modos de execução: usuário e *kernel*. O modo usuário é o modo padrão para as aplicações e o modo *kernel* é usado quando há necessidade de serviços mais privilegiados, como a escrita em disco. O KVM acrescenta um terceiro modo, o convidado. As máquinas virtuais (MV) criadas pelo KVM são processos sendo executados em modo convidado. Com o dispositivo /dev/kvm, uma MV tem seu próprio espaço de endereço separado do espaço do *kernel* ou de qualquer outra MV em execução. Uma máquina virtual KVM é tratada como um processo normal, e pode ser morto como qualquer outro processo.

O KVM utiliza a virtualização de *hardware* para realizar a virtualização de processos. Cada máquina virtual possui um *hardware* virtualizado: uma placa de rede, disco, placa gráfica e outros. O gerenciamento de memória é tratado dentro do *kernel* através do dispositivo /dev/kvm, a memória física é mapeada para o sistema operacional convidado. Uma tabela de páginas de memória é mantida para realizar a conversão de endereços físicos convidados para endereços físicos hospedeiros. Requisições de E/S são virtualizadas através de um processo QEMU levemente modificado (assunto da próxima subseção) (HABIB, 2008).

Uma das limitações do KVM é que ele funciona apenas em processadores que possuem suporte à virtualização, deixando de fora a maioria das máquinas antigas (máquinas com processadores diferentes de Intel-VT e AMD-V) (ARBIZA, 2010). Como o KVM utiliza virtualização completa, baseado em partes do QEMU, é possível rodar Windows ou qualquer outro sistema operacional, sem necessidade de modificação especial. Uma outra característica importante é que o *software* KVM também é *open source*, sob licença GPL.

3.3 QEMU

O QEMU é uma ferramenta de código aberto, com licença GPL, que pode ser utilizada tanto como emulador de máquina ou como *software* de virtualização. Quando utilizado como emulador, ele permite que os aplicativos feitos para serem executados em um sistema operacional sejam executados em outro. Apesar da utilização da técnica de emulação introduzir um *overhead* no sistema, o QEMU possui um desempenho muito bom, por utilizar a técnica de tradução dinâmica, ou seja, o emulador converte partes do código para que o processador execute o conjunto de instruções. Como emulador, o QEMU pode realizar a emulação total do sistema ou a emulação no modo usuário.

Na emulação total do sistema, ele realiza a emulação do *hardware* de forma completa, ou seja, processador e periféricos. O emulador é utilizado para rodar os diferentes sistemas operacionais. Já no modo usuário a emulação está disponível somente para sistemas Linux, onde processos Linux compilados poder ser executados em outra plataforma (SOUZA, 2009). Sendo utilizado como *software* virtualizador, o QEMU possui um bom desempenho graças a um *driver* de aceleração, KQEMU (KQEMU, 2010), que permite que as instruções da máquina virtual sejam executadas diretamente no *hardware* do processador físico.

3.4 VirtualBox

O VirtualBox é uma ferramenta de código aberto, também com licença GPL, desenvolvido para realizar a virtualização de arquiteturas x86. Pode ser utilizado em ambientes Linux, Windows, OpenSolaris e Macintosh. O VirtualBox foi desenvolvido para ser utilizado em *desktops*, servidores ou embarcados (ALKMIM,2009).

Abaixo são listados alguns recursos disponíveis no VirtualBox (VIRTUALBOX, 2010):

Modularidade: o VirtualBox possui um desenho modular com interfaces de programação interna bem definidas e um desenho cliente/servidor. Isso torna o seu controle mais fácil.

Descrição da máquina virtual em XML: as definições de configuração das máquinas virtuais são armazenados em XML e são independentes das máquinas locais, tornando possível e fácil a transferência das definições para outros computadores.

***Guest Additions* para Windows e Linux**: o VirtualBox possui um *software* especial que pode ser instalado dentro das máquinas virtuais Windows e Linux para melhorar o desempenho e fazer integração de forma mais simples.

Pastas compartilhadas: o VirtualBox permite o compartilhamento de pastas, ou seja, as pastas das máquinas hospedeiras podem ser acessadas pelas máquinas hóspedes.

Controladores USB virtuais: dispositivos USB podem ser ligados às máquinas virtuais sem a necessidade de instalar *drivers* adicionais.

Uma questão que paira atualmente sobre o VirtualBox é se este continuará sendo apoiado como *software livre* pela Oracle, após a compra da Sun. Ainda não há uma posição clara a esse respeito, o que deixa a comunidade *software livre* com várias apreensões a respeito.

3.5 Desempenho de Hipervisores Livres

Existem vários trabalhos dedicados à avaliação de desempenho de máquinas virtuais. Alguns como (SANTOS, 2005) apresentam uma análise do desempenho global em ambientes virtualizados e uma metodologia para efetuar a consolidação de servidores. Foi realizada a análise de desempenho do ambiente virtual e do ambiente real utilizado como hospedeiro. Os aspectos considerados foram processador, sistema de memória, discos rígidos e placas gráficas. Como esperado, o desempenho do ambiente real foi superior ao desempenho do ambiente virtual, dentre as máquinas virtuais criadas tanto em ambiente Linux como em Windows as máquinas VMware mostraram melhor desempenho. Em relação à consolidação de servidores foi criada uma metodologia com todas as etapas necessárias para consolidação de vários servidores físicos em um único servidor.

Um outro exemplo é (SANTOS, 2008) que realiza uma comparação de desempenho entre o KVM e as abordagens utilizadas pelo Xen, analisando características como acesso à memória, disco e utilização do processador. Dentre os aspectos analisados, a paravirtualização utilizada pelo Xen se apresentou como uma solução mais eficaz e com baixo impacto sobre o desempenho de tais dispositivos. Porém, devido às modificações necessárias para execução dos sistemas operacionais, essa técnica não permite executar sistemas cujo *kernel* não possa ser alterado. Para tais sistemas, a solução é a utilização de outras técnicas de virtualização, mesmo que isso cause uma perda de desempenho em determinados dispositivos.

Existem outros estudos que apesar de tratar de desempenho de máquinas virtuais, focam em uma ferramenta específica, como é o caso de (REIS, 2008), que apresenta um estudo sobre o gerenciamento de recursos do monitor de máquinas virtuais Xen. Neste estudo, foram destacados o isolamento dos

domínios e a migração de máquinas virtuais entre diferentes redes de computadores. A técnica de virtualização utilizada pelo Xen foi apresentada em detalhes para que se entendesse como é possível garantir um desempenho tão próximo a ambientes não virtualizados.

Em (SANTOS, 2008) é apresentado um estudo sobre o *VEPMon*, uma ferramenta para monitorar o desempenho de máquinas virtuais em execução no ambiente virtual Xen. O *VEPMon* está disponível somente para ambiente virtual Xen e para ambientes Linux, em modo texto, sendo útil para o monitoramento de serviços *Web* virtualizados que necessitem de planejamento de alocação de recursos para o ambiente virtual. Como trabalho futuro podemos citar um aperfeiçoamento da ferramenta *VEPMon* para monitorar outros ambientes virtuais além de Xen.

Verifica-se ainda um grande número de estudos na área de virtualização e segurança, exemplos são (LAUREANO, 2003) e (LAUREANO, 2004), em que máquinas virtuais foram utilizadas para detectar e combater ataques a serviços de rede, através da implantação de sistemas detectores de intrusão baseados em *host*. Em ambos estudos o protótipo construído foi satisfatório e a utilidade das máquinas virtuais mais uma vez apresentada com êxito.

São poucos os artigos que tratam somente sobre o desempenho de rede em máquinas virtuais, a seguir são apresentados dois desses trabalhos. Em (URCSHEI, 2007), é apresentado um estudo sobre a influência do *overhead* introduzido pela camada de virtualização no desempenho de rede. O trabalho realiza a avaliação de desempenho do Xen, a métrica utilizada é a vazão (Mbps) obtida em comunicações TCP/IP, medida com o *benchmark Netperf*. Com este estudo foi possível analisar que ao se utilizar máquinas virtuais em aplicações que exijam um desempenho de rede adequado deve-se escolher adequadamente as configurações de parâmetros como tamanho de *buffer* e MTU.

Em (SCHMIDT, 2007), por sua vez, é realizado uma análise de desempenho da vitualização de rede nos sistemas Xen e OpenVZ (OPENVZ, 2010), que apresentam diferentes abordagens e constituem soluções populares de virtualização. Observou-se que OpenVZ obteve um desempenho de rede superior ao Xen, principalmente para mensagens pequenas e médias. Isso se explica, pois o Xen utiliza a técnica de paravirtualização, em que são necessárias mudanças no *kernel* do sistema operacional hóspede. Já o OpenVZ utiliza a virtualização no nível do sistema operacional, sendo utilizado o mesmo *kernel* para executar os ambientes no hospedeiro. Observou-se ainda que para certos tamanhos de pacotes o OpenVZ consegue atingir uma taxa de transmissão muito próxima a um sistema não virtualizado. Sendo este um ponto favorável para a escolha do mesmo em sistemas virtualizados onde o desempenho de rede é um ponto crítico.

Este trabalho procura complementar os estudos de análise de desempenho de hipervisores por meio da avaliação de: Xen, QEMU e VirtualBox, quanto ao desempenho de rede. O estudo do hipervisor KVM não foi possível devido ao fato de o mesmo funcionar apenas em processadores que possuem suporte à virtualização, sendo esta uma das limitações do ambiente configurado para os testes. Optou-se ainda pela não utilização do software OpenVZ, por este ser um virtualizador em nível de sistema operacional, que utiliza apenas um kernel. Com isso, o sistema operacional convidado é executado como um outro processo qualquer sendo executado sobre o sistema operacional hospedeiro, o que foge dos objetivos do trabalho, que é analisar o desempenho de rede um sistema computacional depois de inserida uma camada de virtualização sobre o mesmo.

4 METODOLOGIA

O presente trabalho trata-se de um estudo operacional, pois visa encontrar o melhor resultado através da análise de dados. Nesta perspectiva, realizou-se o estudo de ferramentas de virtualização com foco no desempenho de rede das mesmas.

Quanto à sua natureza, o estudo se classifica como pesquisa fundamental cujo objetivo é entender ou descobrir novos fenômenos com foco em conhecimentos básicos e fundamentais (ZAMBALDE *et al* 2008).

Quanto aos objetivos, a pesquisa se classifica como descritiva que tem por finalidade a observação, registro e análise de fenômenos ou sistemas técnicos (ZAMBALDE *et al* 2008). Para tanto, o método de coleta de dados utilizado no trabalho em questão foi a análise de sistemas técnicos através do *benchmark* de rede *Iperf*. O estudo se caracteriza como pesquisa laboratorial, uma vez que ocorreu num ambiente previamente estabelecido, de sistemas computacionais ligados em rede local.

Os procedimentos utilizados na realização dos estudos, bem como o detalhamento das abordagens acima citadas, serão descritos na seção 4.1.

4.1 Procedimentos e métodos

Os testes realizados tem como principal objetivo analisar e comparar o desempenho de rede de três hipervisores de virtualização: Xen, VirtualBox e QEMU. Para tanto, foram configurados dois ambientes de testes para cada hipervisor. As máquinas virtuais criadas possuíam as mesmas configurações e aplicações, sendo apenas adaptadas ao hipervisor e ao ambiente em questão. Cada máquina virtual foi executada com uma CPU virtual, HD de 10GB, 128

MB de memória RAM e placa de rede em modo *bridge*, interface virtual de uso comum que une fisicamente várias interfaces, virtuais e real.

A distribuição Linux adotada nas máquinas virtuais envolvidas nos testes foi o Debian, por ser voltado para utilização em servidores, facilitando assim, o objetivo do estudo. Em cada ambiente, foram realizados 10 testes para cada hipervisor, sendo que, a máquina virtual em questão foi a servidora e uma máquina real a cliente.

A ferramenta de *benchmark* utilizada para avaliar o desempenho de rede dos hipervisores foi o *iperf*, *software* livre mantido pela Universidade de Illinois. Com o *iperf* é possível medir largura de banda, perda de pacotes na rede, atraso, variação de atraso e MTU (Unidade Máxima de Transmissão). É um programa cliente/servidor adequado para medições ativas, podendo ser utilizado com o protocolo UDP ou TCP, além disso, é possível também lidar com múltiplas conexões simultâneas.

Quando utilizado com o protocolo UDP é possível realizar a medição de largura de banda, atraso e perda de pacotes. Utilizando pacotes TCP, o *iperf* mede quantidade máxima de tráfego que é possível alcançar entre dois pontos finais, há um mecanismo de controle de congestionamento que procurará usufruir toda banda disponível no caminho.

Para realização dos testes em questão, foi utilizado o protocolo TCP, a fim de medir o tráfego máximo alcançável entre as máquinas analisadas. Para isso, foi disparado o comando **iperf -s** na máquina virtual (servidora), e **iperf -c <IP_MAQUINA_VIRTUAL>** na máquina real (cliente).

No ambiente 1, a máquina virtual foi executada em um computador *desktop*, com sistema operacional Gentoo (atualizado), a máquina cliente foi um *notebook*. Os hipervisores analisados foram Xen, com *kernel* 2.6.32-r1 e hipervisor 3.4.2-r1, QEMU 0.11.0 e VirtualBox 3.0.12. A seguir estão as configurações das máquinas utilizadas no ambiente de testes 1 :

- Máquina Hospedeira
    - Processador: AMD Athlon(tm) 64 X2 Dual Core
    - 3 GB de memória RAM
    - HD sata 160 GB
    - Placa de rede: Broadcom Corporation NetXtreme BCM5754 Gigabit Ethernet

- Máquina Cliente (*Notebook*)
    - Processador: Intel Core 2 Duo
    - 3 GB de memória RAM
    - HD sata 120 GB
    - Placa de rede: Broadcom Corporation NetLink BCM5784M Gigabit Ethernet

Segue abaixo, a Figura 4.1, que representa a estrutura do ambiente 1.

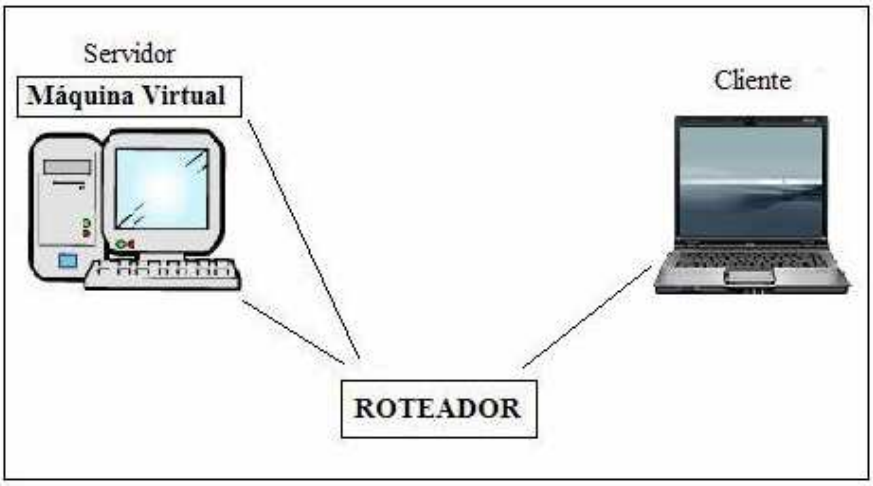

Figura 4.1: Ambiente de testes 1.

No ambiente 2, a máquina virtual foi executada em um *notebook*, com sistema operacional Debian 5.01, e a máquina cliente foi um computador *desktop*. Os hipervisores analisados foram Xen com *kernel* 2.6.26 -1 e hipervisor 3.2.-1, QEMU 0.9.1 e VirtualBox 3.1.6.

Pode-se observar, que foram utilizadas diferentes versões dos hipervisores em estudo em cada ambiente, contudo verifica-se que não houve alteração nas tecnologias de virtualização referentes a acesso a dispositivos de rede dos mesmos. Verifica-se ainda que a escolha de diferentes versões para os hipervisores não impactaram nos resultados obtidos, como será apresentado no seguinte capítulo. A seguir estão as configurações das máquinas utilizadas durante os testes no ambiente 2:

- Máquina Hospedeira (*Notebook*)
    - Processador: Intel Core 2 Duo
    - 3 GB de memória RAM
    - HD sata 120 GB
    - Placa de rede: Broadcom Corporation NetLink BCM5784M Gigabit Ethernet

- Máquina Cliente
    - Processador: Intel Pentium 4
    - 512 MB de memória RAM
    - HD ide 20 GB
    - Placa de rede: Intel Corporation 82540EM Gigabit Ethernet

Segue abaixo, a Figura 5.2, que representa a estrutura do ambiente 2.

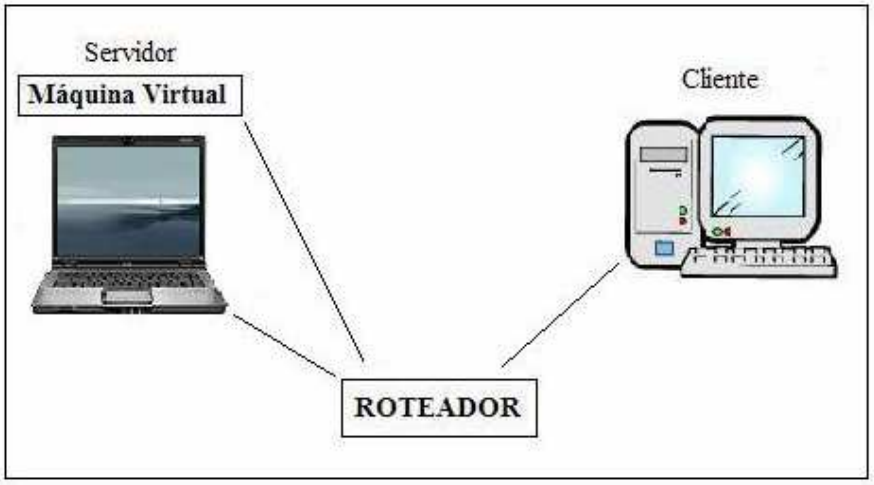

Figura 4.2: Ambiente de testes 2.

Em cada ambiente foi criada uma máquina virtual para cada hipervisor analisado. A máquina virtual foi iniciada e logo em seguida disparado em seu terminal o comando **iperf –s,** sendo esta a partir de então a máquina servidora (Figura 4.3). A partir deste momento o *iperf* ficará executando conexões vindas de outros computadores.

```
virtual:~# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

Figura 4.3: Comando *iperf* na máquina servidora

Na outra ponta do link uma máquina real foi conectada à máquina virtual servidora através do comando **iperf –c <IP_MV>**. A partir desse momento por 10 segundos as duas instâncias mandaram o máximo de dados

possível para utilizar a capacidade máxima de transmissão do canal. As saídas do comando no servidor e no cliente são bem semelhantes. Nas Figuras 4.4 e 4.5, respectivamente, podem ser observadas as saídas do comando na máquina cliente e na máquina servidora. No teste abaixo, a velocidade final conseguida foi de aproximadamente 44 Mbps.

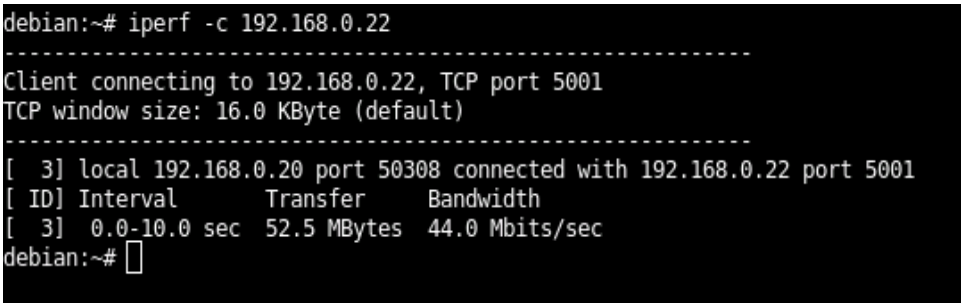

Figura 4.4: Saída do comando **iperf** na máquina cliente.

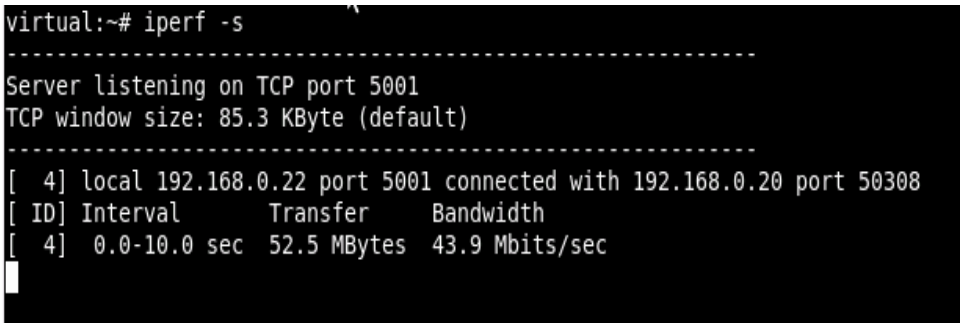

Figura 4.5: Saída do comando **iperf** na máquina servidora.

Analisou-se ainda o desempenho de rede de um sistema não virtualizado. Para tanto, utilizou-se os mesmos testes e os mesmos ambientes de testes descritos acima, porém, neste caso, a máquina hospedeira (sem a camada de virtualização) foi utilizada como máquina servidora, e a máquina utilizada

como cliente em cada ambiente foi a mesma. Para tal análise, realizou-se também 10 testes para cada ambiente em questão, a fim de verificar a impacto da camada de virtualização quando analisado o desempenho de rede em sistemas virtualizados.

5 RESULTADOS E DISCUSSÃO

Ao longo deste capítulo será apresentada a análise dos resultados obtidos com os testes aplicados em cada hipervisor, bem como uma breve comparação entre os mesmos. Serão apresentados também os resultados obtidos quando analisado o sistema não virtualizado, verificando assim a influência da camada de virtualização no desempenho de rede dos mesmos. Como descrito na seção anterior, foram realizados 10 testes para cada hipervisor e 10 testes para o sistema não virtualizado em cada ambiente computacional.

5.1 Ambiente de testes 1

Os resultados obtidos no ambiente de testes 1, quando analisados somente os sistemas virtualizados, pode ser observado no gráfico mostrado na Figura 5.1.

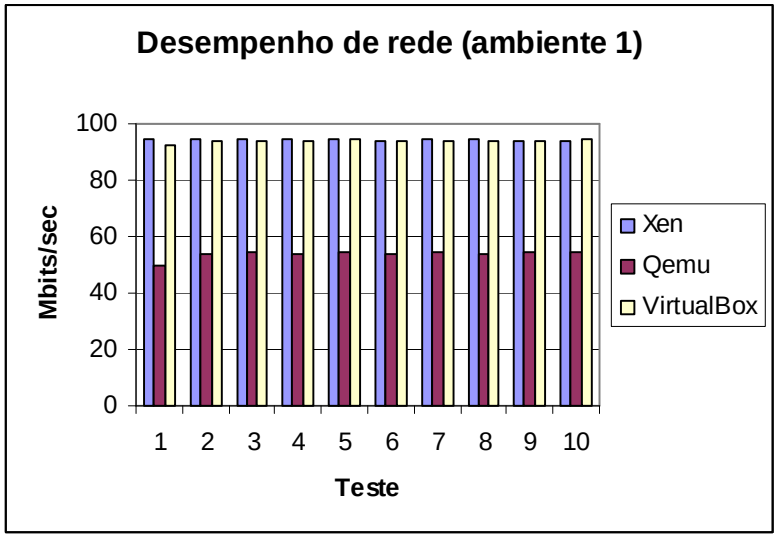

Figura 5.1: Resultados obtidos no ambiente 1.

Analisando a Figura 5.1, pode-se observar uma superioridade no desempenho de rede dos hipervisores Xen e VirtualBox quando comparados com o hipervisor QEMU. Xen e VirtualBox chegam a atingir praticamente o dobro do desempenho obtido com QEMU. Este fato pode ser explicado pela técnica de virtualização adotada no mesmo. Como mencionado anteriormente, a técnica de emulação de hardware pode introduzir um *overhead* no sistema, o que pode explicar essa notável diferença de desempenho de rede.

Observando ainda a Figura 5.1, Xen e VirtualBox não apresentaram diferenças significativas quando comparados entre si, o que mostra que VirtualBox se encontra em boa fase de desenvolvimento, podendo até chegar ao mesmo desempenho de rede de um sistema paravirtualizado. Cabe esclarecer ainda que o VirtualBox surgiu no mercado posteriormente ao Xen.

5.2 Ambiente de testes 2

Os resultados obtidos no ambiente de testes 2, quando analisados somente os sistemas virtualizados, pode ser observado no gráfico mostrado na Figura 5.2.

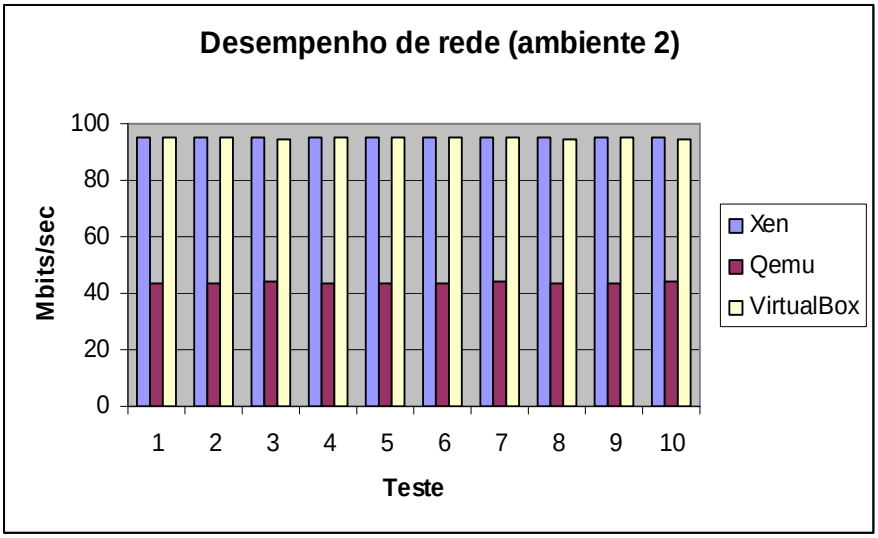

Figura 5.2: Resultados obtidos no ambiente 2.

Analisando a Figura 5.2, pode-se observar que o desempenho de rede do hipervisor QEMU chega a ser praticamente duas vezes pior quando comparado com Xen e Virtualbox, isso pode ser explicado pela técnica de virtualização adotada pelo mesmo, como mencionado, a emulação de *hardware* introduz um *overhead* no sistema, podendo ser a causa dessa notável diferença de desempenho.

Quando comparados Xen e VirtualBox, pode-se perceber que não existem diferenças significativas nos resultados obtidos, o que mostra que VirtualBox está em boa posição de desenvolvimento e melhorias.

5.3 Sistema não virtualizado *versus* sistemas virtualizados

A seguir, serão apresentados os resultados obtidos quando analisados além de sistemas virtualizados, os sistemas não virtualizados (sistema real), é

importante lembrar que em cada ambiente, o sistema real analisado foi sempre o sistema que serviu de hospedeiro para as máquinas virtuais, é importante salientar também que, nos testes realizados tais sistemas foram os servidores.

Com o objetivo de verificar qual a influência da camada de virtualização no aspecto analisado, pode-se observar os resultados obtidos nas Figura 5.3 e Figura 5.4.

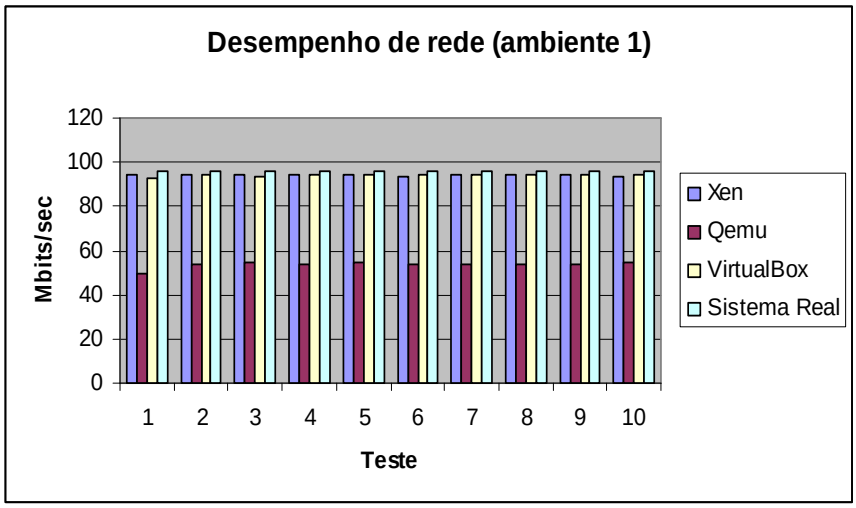

Figura 5.3: Resultados obtidos no ambiente 1, com sistema real.

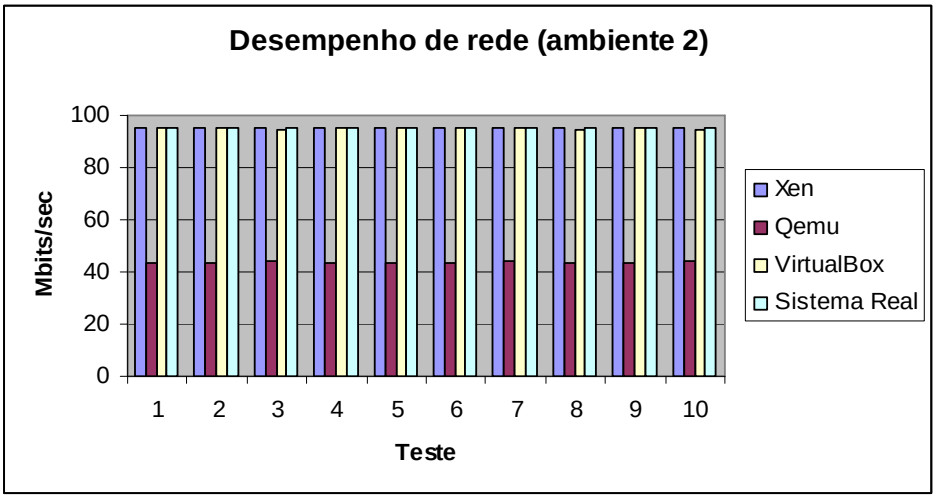

Figura 5.4: Resultados obtidos no ambiente 2, com sistema real.

Analisando as Figuras 5.3 e 5.4, pode-se observar que o desempenho de rede dos hipervisores Xen e VirtualBox foi muito próximo ou praticamente o mesmo de um sistema não virtualizado (sistema real), o que mostra que a introdução da camada de virtualização de tais hipervisores nos sistemas quase não introduziu *overheads* quando analisado o desempenho de rede dos mesmos.

Observando ainda as Figuras 5.3 e 5.4, verifica-se que o hipervisor QEMU apresentou um desempenho de rede que chega a ser praticamente a metade do desempenho de um sistema não virtualizado (sistema real), como já explicado anteriormente, a técnica de emulação utilizada por tal hipervisor pode ser a causa dessa diferença de resultados.

5.4 Ambiente 1 *versus* Ambiente 2

Quando analisados os impactos dos ambientes de testes nos resultados obtidos, pode-se observar que sua influência foi praticamente nula, como

mostrado na Figura 5.5, que apresenta as médias aritméticas dos resultados em cada ambiente de testes.

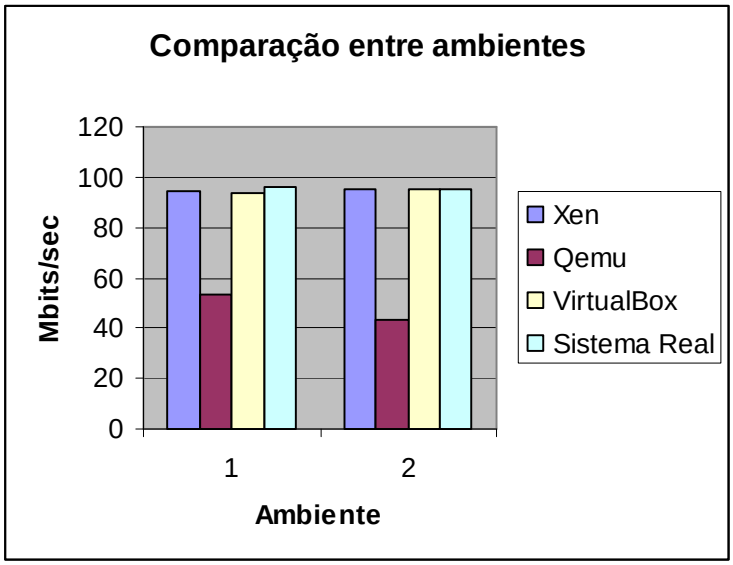

Figura 5.5: Comparação entre ambientes de testes.

Observando ainda a Figura 5.3, verifica-se uma pequena diferença encontrada com o hipervisor QEMU em relação aos ambientes de testes. Porém tais diferenças não impactaram nos resultados práticos obtidos, já que QEMU teve sempre o pior desempenho dentre os três hipervisores analisados. Estas pequenas diferenças de resultados podem ser explicadas pela utilização de um melhor hardware no ambiente de testes 1, já que para QEMU este é um item significante devido a técnica de emulação de *hardware* utilizada pelo mesmo.

5.5 Discussão

Para o hipervisor QEMU, é possível a utilização de módulos distintos de aceleração, o KQEMU ou KVM, porém para utilização do último é necessário processadores com suporte à virtualização, uma limitação do ambiente de teste utilizado. Sendo assim, foi adotada para o mesmo, a técnica de emulação de *hardware* com o módulo KQEMU. Como descrito anteriormente, o baixo rendimento do hipervisor em questão pode ser explicado pela técnica de virtualização empregada. Cabe, portanto, a trabalhos futuros a análise da influência do módulo utilizado quando se trata de desempenho de rede.

Os resultados obtidos com os hipervisores Xen e VirtualBox foram praticamente idênticos, sendo que os mesmos deixaram o VirtualBox em boa posição em relação a desempenho de rede. Neste contexto, vale ressaltar que o Xen foi desenvolvido voltado para virtualização de servidores, utilizando a técnica da paravirtualização a fim de se obter praticamente um mesmo desempenho de uma máquina real. Por outro lado, o VirtualBox foi desenvolvido para virtualização de desktops, com uma interface amigável mostrando preocupação com a satisfação do usuário final.

Analisando, portanto, os objetivos de cada projeto de software, é inevitável mencionar o ótimo resultado obtido com o VirtualBox em relação à desempenho de rede. Para trabalhos futuros propõe-se a comparação de VirtualBox e Xen em relação a outros aspectos como processador, sistema de memória, discos rígidos, placas gráficas e outros.

6 CONCLUSÕES

Durante a execução do trabalho pôde-se verificar a crescente utilização de técnicas de virtualização. A evolução dos recursos computacionais possibilitou o aperfeiçoamento da técnica, sendo assim, diversos métodos e ferramentas têm sido propostos, fazendo com que a escolha de um deles não seja trivial. Dentre os fatores a se considerar, o desempenho de rede é fundamental já que os computadores atuais dependem fortemente da comunicação entre si. Nessa perspectiva, a comparação de desempenho de rede entre ferramentas de virtualização torna-se viável para auxílio da escolha da tecnologia a ser utilizada.

Percebeu-se que, os ambientes configurados para os testes não influenciaram nos resultados dos mesmos. Além disso, o hipervisor VirtualBox apresentou um resultado bastante satisfatório, chegando a um desempenho praticamente igual ao hipervisor Xen, que utiliza a paravirtualização. O desempenho de rede do hipervisor QEMU, chegou a ser duas vezes pior que Xen e VirtualBox, o que já seria esperado devido à sua abordagem de virtualização.

Quando analisados os impactos das camadas de virtualização introduzidas por cada hipervisor, verificou-se que Xen e VirtualBox apresentaram um ótimo resultado, já que ambos mostraram praticamente o mesmo desempenho de um sistema não virtualizado, mostrando que, o objetivo da virtualização de criar sistemas virtuais com execução idêntica à sistemas reais pode ser alcançado. Em relação ao QEMU, o *overhead* introduzido por sua camada de virtualização pode ser explicado pela técnica utilizada pelo mesmo.

Contudo, o trabalho apresenta notáveis limitações, por se tratar de um estudo de apenas três hipervisores em meio a tantas tecnologias propostas. Sendo assim, recomenda-se para trabalhos futuros, a comparação de outras tecnologias de virtualização e ampliação dos testes aqui apresentados. Isso se

justifica, inclusive, que o avanço das técnicas de virtualização pode propiciar a mudança no comportamento de rede dos hipervisores analisados.

REFERENCIAL BIBLIOGRÁFICO

ALKMIM, G., P., **Migração de Máquinas Virtuais utilizando Xen, GFS e ConVirt.** Universidade Federal de Lavras, 2009.

ARBIZA, L. M., KREUTZ, D. L., **Apache2 e MySQL 5 sobre Máquinas Virtuais Xen e KVM**, Universidade Federal do Pampa, 2010.

BARHAM, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A., Xen and the Art of Virtualization. **In Proceedings of the nineteenth ACM symposium on Operating systems principles**, pág 164-177, 2003.

BOCHS, **Bochs IA-32 Emulator Project.** Disponível em http://bochs.sourceforge.net/, acessado em maio de 2010, 2010.

HABIB, I. (2008**). Virtualization with KVM**. *Linux Journal*. Disponível em: http://www.linuxjournal.com/article/9764. Acesso em: maio de 2010.

HUANG, W., Liu J., Abali, B., Koop, M., Panda, D., Nomad: Migrating OS-bypass Networks in Virtual Machines. **In Proceedings of the 3rd international conference on Virtual execution environments**, pág 158-168, 2007.

JONES, M.T., **Virtual Linux: An overview of virtualization methods, architectures, and implementations**. Disponível em: http://www.ibm.com/developerworks/library/llinuxvirt/**.** Consultado em: maio de 2010.

KQEMU, **QEMU Accelerator**. Disponível em http://wiki.qemu.org/KQemu/Doc/, acessado em maio de 2010.

KVM, **Kernel-based Virtual Machine.** Disponível em http://www.linux-kvm.org, acessado em maio de 2010, 2010.

LAUREANO, M. A. P., **Detecção De Intrusão Em Máquinas Virtuais.** Ph. D. Thesis, Pontifícia Universidade Católica do Paraná, 2003.

LAUREANO, M. A. P., **Uma Abordagem para a Proteção de Detectores de Intrusão Baseada em Máquinas Virtuais.** Ph. D. Thesis, Pontifícia Universidade Católica do Paraná, 2004.

LAUREANO, M., In Novatec (Editors), **Máquinas Virtuais e Emuladores. Conceitos, Técnicas e Aplicações**, Novatec, 2006.

LAUREANO, Marcos A. e MAZIERO, Carlos A. **Virtualização: Conceitos e Aplicações em Segurança –** Pontifícia Universidade Católica do Paraná, 2008.

LGUEST, **Lguest Hypervisor**. Disponível em http://lguest.ozlabs.org//, acessado em maio de 2010, 2010.

OPENVZ. **OpenVZ**. Disponível em http://www.openvz.org/, acessado em maio de 2010, 2010.

POPEK, Gerald J. e GOLDBERG, Robert P. Formal Requirements for Virtualizable ThirdGeneration Architectures**. Communications of the ACM**. Volume 17, Número 7, Páginas 412-421, Julho de 1974.

QEMU, **QEMU Processor Emulator**. Disponível em http://www.qemu.org, acessado em maio de 2010, 2010.

REIS, V., Q., Cerqueira, R., F., G., **Gerenciamento de Recursos no Monitor de Máquinas Virtuais Xen.** Pontifícia Universidade Católica do Rio de Janeiro, 2008.

SANTOS, G., M., **Máquinas Virtuais: Avaliação de Desempenho e Consolidação de Servidores.** Universidade Luterana do Brasil, 2005.

SANTOS, M., Fernandes, C., Benevenuto, F., Almeida, V., Almeida, J., **VEPMon: Uma Ferramenta de Monitoração de Desempenho para Ambientes Virtuais.** Universidade Federal de Minas Gerais (UFMG), 2008.

SANTOS, R. C. M., **Arquiteturas virtualizadas, comparação entre Xen e KVM.** Universidade Federal de Santa Maria (UFSM), 2008.

SANTOS, R. C. M., CHARÃO, A. S., **Análise Comparativa de Desempenho do Hipervisor Xen: Paravirtualização *versus* Virtualização Total.** Universidade Federal de Santa Maria (UFSM), 2008.

SCHMIDT, A. H., BOUFLEUR, M. P., SANTOS, R. C. M., CHARÃO, A.S. **Análise de Desempenho da Virtualização de Rede nos Sistemas Xen e OpenVZ**. Universidade Federal de Santa Maria (UFSM) – RS, 2007.

SILVA, R. F., **Virtualização de Sistemas Operacionais**, Instituto Superior de Tecnologia em Ciência da Computação (ISTCC), 2007.

SOUZA, C. A., LIMA, I. L. C., ALBUQUERQUE, T. O., **Expansão do portfólio de virtualização do OurGrid,** Universidade Federal de Campina Grande, 2009.

SOUZA, F. B. **Uma arquitetura para Monitoramento e Medição de Desempenho para Ambientes Virtuais**. Ph. D. Thesis, Universidade Federal de Minas Gerais, 2006.

URSCHEI, F., Pelegrini, E. J., Silva, M. A. L., Midorikawa, E. T., Carvalho, T. C. M. B. Análise Multiparamétrica do *Overhead* de Rede em Máquinas Virtuais. **In Proceedings of the IV Workshop de Sistemas Operacionais**, pág 816-827, 2007.

VIRTUALBOX. **VirtualBox**. Disponível em http://www.virtualbox.org/, acessado em maio de 2010, 2010.

VMWARE, **VMware**. Disponível em http://www.vmware.com/, acessado em maio de 2010, 2010.

WLODARZ, J. J. **Virtualization: A double-edged sword**. Disponível em http://arxiv.org/PS_cache/arxiv/pdf/0705/0705.2786v1.pdf, acessado em maio 2010, 2007.

XEN, **Xen Hypervisor**. Disponível em http://www.xen.org/, acessado em maio de 2010, 2010.

ZAMBALDE, A. L.; PÁDUA, C. I. P. S.; ALVES, R. M. **O documento científico em Ciência da computação e Sistemas de Informação**. Lavras/MG: DCC/UFLA, 2008.