UNIVERSIDADE DE CAXIAS DO SUL

Centro de Ciências Exatas e Tecnologia

Curso de Bacharelado em Ciência da Computação

Yuri Franzoni da Silva

Uma Avaliação sobre a Viabilidade do Uso de Técnicas de Virtualização em Ambientes de Alto Desempenho

Caxias do Sul

2009

Yuri Franzoni da Silva

Uma Avaliação sobre a Viabilidade do Uso de Técnicas de Virtualização em Ambientes de Alto Desempenho

Trabalho de Conclusão de Curso para obtenção do Grau de Bacharel em Ciência da Computação da Universidade de Caxias do Sul.

André Luís Martinotto

Orientador

Caxias do Sul

2009

Agradecimentos

Quero agradecer, primeiro, a Deus por ter me dado todas as condições para chegar até aqui.

Aos meus pais, Maira e Valter, por todo o suporte, carinho e paciência que me dispuseram durante esta caminhada, nas horas boas e também nas horas mais complicadas.

Ao meu irmão, Andrei, por todos os conselhos e comentários, às vezes nem tão amigáveis, mas sempre construtivos.

Aos meus sogros, Ana Lúcia e Pedro, pela paciência e pelo carinho de aturar um genro nem sempre bem-humorado.

Aos meus amigos, que compartilharam comigo cada dia dentro da UCS (e também de outras instituições, como poderia não lembrar deles?).

Ao meu professor orientador, por todo o conhecimento, disponibilidade e suporte prestados durante todo o tempo em que convivemos na Universidade.

À equipe da empresa onde trabalho, especialmente à Luciana, pela força que me deram quando mais precisei, além da paciência de conviver com uma pessoa com a cabeça em vários lugares ao mesmo tempo.

Ao Jerônimo do NPD, que deu dicas valiosas sobre algumas ferramentas estudadas.

A todos aqueles que, de uma forma ou outra, contribuíram para que este momento (finalmente) chegasse.

E por último, mas não menos importante – muito longe disso –, à Raquel, que tem dividido comigo meus melhores e piores momentos já há alguns anos e que nunca hesitou em me ajudar quando precisei, da forma que fosse necessário... que soube me acalmar quando tudo estava complicado e soube também comemorar a cada batalha vencida. Sem o teu apoio, eu não teria chegado aonde cheguei.

Resumo

As técnicas de virtualização, apesar de não serem tecnologias novas, ocupam lugar de destaque no atual cenário de Tecnologia da Informação. Através dessas técnicas, é possível executar diversas máquinas virtuais sobre um computador, proporcionando assim uma série de vantagens no que se refere à disponibilidade, segurança e aproveitamento de recursos físicos do equipamento, ao provável custo de alguma perda de desempenho.

Neste trabalho é feito um estudo sobre a aplicabilidade das técnicas de virtualização em ambientes de alto desempenho. Para tanto, foram estudadas as técnicas de implementação de máquinas virtuais existentes e diversas ferramentas de virtualização disponíveis no mercado, bem como algumas metodologias de *benchmark* para avaliação do desempenho de cada ferramenta. Cada software foi submetido a uma série de testes para avaliação de performance sob diversos aspectos como eficiência na utilização de CPU, memória, I/O e comunicação de rede, de forma a determinar a viabilidade da aplicação dessas técnicas em um ambiente de alto desempenho.

Palavras-chave: virtualização, máquinas virtuais, *benchmarks*, ambiente de alto desempenho

Abstract

The virtualization techniques, although not new technologies, occupy a prominent place in the current Information Technology scenario. Through these techniques, it is possible to run several virtual machines on one computer, thus providing a number of advantages in terms of availability, security and utilization of the equipment's physical resources, with the likely cost of some performance loss.

In this paper, we study the applicability of virtualization techniques in high-performance computing environments. To achieve this goal, the current virtual machine implementation techniques and some of the virtualization tools available on the market have been studied, as well as some benchmark methodologies for assessing the performance of each tool. Each software has undergone a series of tests to evaluate performance in several respects, such as utilization rates of CPU, memory, I/O and network bandwidth, in order to determine the feasibility of applying these techniques in a high-performance computing environment.

Keywords: virtualization, virtual machines, benchmarks, high-performance computing

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AMD	*Advanced Micro Devices*
API	*Application Program Interface*
ATA	*Advanced Technology Attachment*
ATI	*ATI Technologies*
BSD	*Berkeley Software Distribution*
CPU	*Central Processing Unit*
DDR2	*Double Data Rate 2*
DLPI	*Data Link Provider Interface*
ERP	*Enterprise Resource Planning*
FLOP	*Floating-point operations per second*
FPU	*Floating-Point Unit*
GB	*Gigabytes*
GFLOP	*Gigaflop*
GHz	*Gigahertz*
GNU	*GNU is Not Unix*
GPL	*General Public License*
HBA	*Host Bus Adapter*
I/O	*Input/Output*
IBM	*International Business Machines*
IP	*Internet Protocol*
IPv4 e IPv6	*Internet Protocol* versão 4 ou 6
ISA	*Instruction Set Architecture*

JVM	*Java Virtual Machine*
KB	*Kilobytes*
KVM	*Kernel-based Virtual Machines*
MB	*Megabytes*
Mbps	*Megabits* por segundo
MHz	*Megahertz*
MIPS	*Million Instructions Per Second*
NFS	*Network File System*
OLTP	*Online Transaction Processing*
OSDB	*Open Source Database Benchmark*
OSDL	*Open-Source Development Labs*
OSE	*Open Source Edition*
OSI	*Open Systems Interconnection*
OVF	*Open Virtualization Format*
PC	*Personal Computer*
PUEL	*Personal Use and Evaluation License*
QUIPS	*Quality Improvement Per Second*
RAM	*Random-Access Memory*
RDP	*Remote Desktop Protocol*
RPM	Rotações por minuto
SCTP	*Stream Control Transmission Protocol*
SMP	*Symmetric Multiprocessing*
SO	Sistema Operacional
TCO	*Total Cost of Ownership*

TCP	*Transmission Control Protocol*
TI	Tecnologia da Informação
UDP	*User Datagram Protocol*
VM	*Virtual Machine*
VMI	*Virtual Machine Interface*
VMM	*Virtual Machine Monitor*
VRDP	*VirtualBox RDP*

Sumário

1 Introdução

A técnica de virtualização surgiu em meados da década de 60, quando o termo *máquina virtual* foi utilizado pela primeira vez para indicar uma abstração de software de um sistema computacional em hardware. Em 1970, já era comum que cada mainframe tivesse o seu sistema operacional proprietário, ou seja, incompatível com os sistemas que acompanhavam equipamentos de outros fabricantes. Nessa época, o conceito de máquina virtual foi muito explorado, de forma que os fornecedores de hardware passaram a oferecer seus computadores com uma camada de software que oferecia um ambiente completo, similar ao de outro sistema operacional executando diretamente em hardware. Já na década de 80, com a popularização das três principais famílias de sistemas operacionais – Unix, Windows e Macintosh –, o emprego da virtualização acabou por perder uma parte da sua importância (CARISSIMI, 2009).

Entretanto, presenciou-se nos últimos anos um aumento significativo no poder de processamento dos computadores disponíveis no mercado, devido especialmente à popularização de processadores *multi-core*, ou seja, processadores que possuem mais de um núcleo de processamento em um único *chip*. Esse crescimento pode ser notado em máquinas de todos os segmentos, desde os equipamentos mais econômicos até os sistemas mais caros e complexos. Além disso, os preços desses computadores estão cada vez menores, transformando esses equipamentos em uma presença constante em praticamente qualquer ambiente dos nossos dias.

Por outro lado, os sistemas operacionais atuais já não tiram todo o proveito da capacidade de processamento desses equipamentos. Nos dias de hoje, são comuns os relatos de grandes organizações que detém servidores e agregados com taxas de utilização de CPU próximas a 10%. Ou seja, em média, durante 90% do tempo em que estão disponíveis, essas máquinas ficam ociosas. Isso acontece devido ao fato de que esses sistemas passam a maior parte do tempo executando poucas aplicações, quando não somente uma. Isso ocorre devido ao fato de não ser recomendado, nos sistemas operacionais atuais, executar vários serviços e aplicativos com objetivos distintos (por exemplo, servidores de e-mail, banco de dados e *firewall*), devido a incompatibilidades e conflitos que possam surgir da heterogeneidade de versões de sistemas operacionais (NOVELL, 2007).

Nesse contexto, as soluções baseadas em tecnologias de virtualização, tais como Xen e VMware, estão ganhando espaço no mercado, por oferecerem um aproveitamento

mais racional dos recursos de hardware do computador. A utilização de softwares de virtualização torna possível executar cada serviço e aplicativo em um sistema operacional dedicado apenas à execução dessa aplicação e que trabalha sobre uma máquina virtual, para depois executar várias dessas máquinas virtuais simultaneamente em um único computador físico (LAUREANO *et. al*., 2008).

Em sistemas virtualizados, cada máquina virtual é composta por um conjunto de arquivos utilizados pela ferramenta de virtualização, que simulam as condições e recursos de um computador real, tais como memória RAM, disco rígido, placa de vídeo, interfaces de rede e outros. Esse encapsulamento permite gerenciar e efetuar manutenção nessas máquinas virtuais de uma maneira simples, dado que é possível efetuar a restauração do estado de uma máquina virtual com rapidez – basta para isso possuir um *backup* do conjunto de arquivos pertencentes à máquina em questão. Esses *backups*, conhecidos como *snapshots* ou *checkpoints*, são "imagens" fiéis das máquinas virtuais no momento de sua criação (LAUREANO *et. al*., 2008).

Outra vantagem da utilização de virtualização consiste na possibilidade de migração de máquinas virtuais entre computadores físicos, sendo que em determinados casos, não existe nem mesmo a necessidade de parada na execução e a migração ocorre sem grandes perdas de desempenho, facilitando assim o gerenciamento da carga dos equipamentos (VMWARE, 2009a). Por fim, pode-se citar o fato de que em casos em que é necessário reiniciar o sistema, pode-se fazê-lo somente na máquina virtual em questão e isso não irá afetar as demais máquinas (consequentemente, nem seus respectivos serviços) que estejam rodando no computador físico – ao contrário do modelo tradicional, onde todos os serviços precisam ser parados para que o equipamento seja reiniciado por completo.

Não obstante as possibilidades da aplicação da tecnologia de virtualização, é fundamental que sejam analisados também os aspectos negativos dessa abordagem. De modo geral, a maior parte das ferramentas disponíveis no mercado funciona através da implantação de uma camada de software que fica entre o sistema operacional nativo (aquele que é executado pela máquina física) e os sistemas operacionais hóspedes (que executam sobre as máquinas virtuais) (XEN, 2009). Essa camada, por mais eficiente que seja, sempre se traduz em processamento adicional, refletindo no desempenho dos sistemas virtualizados.

O objetivo deste trabalho é efetuar uma avaliação sobre a aplicabilidade de soluções de virtualização em ambientes computacionais de alto desempenho. Para tanto, será feita uma pesquisa das técnicas de virtualização existentes, bem como das principais

ferramentas disponíveis no mercado. Ainda serão descritos os principais métodos de avaliação de performance (*benchmark*) existentes no mercado, além dos pacotes disponíveis para tais testes.

1.1 Estrutura do Trabalho

Este trabalho foi dividido em seis capítulos, detalhados a seguir:

No capítulo 2, “Virtualização”, encontra-se uma breve descrição dos conceitos básicos de virtualização, bem como a classificação das máquinas virtuais segundo sua estratégia de implementação – emulação, máquina virtual de processo e máquina virtual de sistema (virtualização total ou paravirtualização). Também estão descritas três das principais ferramentas de virtualização disponíveis no mercado e que foram utilizadas para testes: Sun VirtualBox, Xen e VMware Server.

No capítulo 3, “Métodos para *Benchmark*”, estão descritos os principais métodos existentes no mercado para avaliação de desempenho de softwares, conhecidos por pacotes de *benchmark*, bem como os objetivos e modo de funcionamento de cada pacote. Ao final deste capítulo, são apresentadas as ferramentas de *benchmarks* que foram utilizadas para avaliação de desempenho das ferramentas de virtualização selecionadas no capítulo anterior.

No capítulo 4, “Testes e Resultados Obtidos”, apresentam-se os resultados dos testes de desempenho que foram executados sobre os softwares de virtualização selecionados. Cada *benchmark* tem seus números detalhados e comentados, de forma a permitir uma comparação entre as ferramentas testadas.

No capítulo 5, “Migração de Máquinas Virtuais com XenServer”, encontram-se os resultados dos testes com a ferramenta Citrix XenServer, a única ferramenta testada que oferece em sua versão gratuita o recurso de migração “quente” de máquinas virtuais (chamada de *XenMotion*). Com esses testes, procurou-se determinar qual a magnitude da perda de desempenho causada pela migração de máquinas virtuais.

Finalmente, no capítulo 6, “Conclusão”, apresentam-se as conclusões finais do trabalho e possíveis contribuições futuras para sua melhoria.

2 Virtualização

Apesar de todo o destaque recebido nos últimos anos, o conceito de virtualização não é novo. Essa técnica possui mais de quarenta anos, sendo que as suas primeiras utilizações remontam aos mainframes das décadas de 60 e 70 (JONES, 2006). Nessa época, a IBM desenvolveu os conceitos de virtualização como uma forma de compartilhar o tempo de processamento de seus equipamentos de alto custo.

Naquela época, era comum que a maioria das organizações pudesse manter um único equipamento de grande porte, que era utilizado tanto para o desenvolvimento quanto para a implantação das aplicações. Porém, desenvolver uma aplicação utilizando o mesmo sistema em que se pretende implantá-la até hoje é considerada uma prática desaconselhável. Isso se deve ao fato de que as atividades de desenvolvimento podem exigir reinicializações do sistema e até mesmo causar instabilidade de forma a interferir no desempenho e na confiabilidade dos aplicativos de produção da empresa. Nesse contexto, as técnicas de virtualização ofereceram a possibilidade de um isolamento dos ambientes de desenvolvimento e de produção (ROSE, 2004).

Além disso, muitas vezes cada *mainframe* tinha seu próprio sistema operacional, até mesmo quando se tratavam de modelos diferentes de um mesmo fabricante. Nesse caso, as máquinas virtuais permitiam que um software legado pudesse ser executado em *mainframes* diferentes. Essa abordagem foi utilizada com sucesso pela IBM na sua linha *System/370* e sucessores, que ofereciam uma máquina virtual portada para várias de suas plataformas, sobre a qual as aplicações eram executadas. Dessa maneira, era possível executar ou migrar uma aplicação entre as plataformas, desde que houvesse uma versão da máquina virtual para a plataforma desejada (CARISSIMI, 2008).

Durante os anos 1980, com a popularização dos computadores e o surgimento dos PCs e dos computadores de pequeno porte, as arquiteturas e os sistemas operacionais convergiram para poucas famílias, sendo que cada uma delas possuía seu conjunto de aplicativos e um público-alvo bem definido, o que diminuiu os problemas até então solucionados pela virtualização (CARISSIMI, 2008). Além disso, a economia proporcionada pelo uso dos PCs acabou por deixar em segundo plano as ideias de virtualização em larga escala. Entretanto, devido ao aumento de capacidade computacional dos processadores atuais, ao forte crescimento dos *data centers* baseados na arquitetura x86 e à disseminação de sistemas distribuídos e das redes de computadores, dentre outros motivos, muitos gerentes de TI estão voltando suas atenções às ideias de virtualização (GOTH, 2007). Neste

capítulo, serão abordados os conceitos básicos sobre virtualização. Além disso, será realizada uma apresentação dos tipos de máquinas virtuais existentes no mercado e a caracterização de cada uma delas.

2.1 O que é a Virtualização

Basicamente, a técnica de virtualização permite a execução de múltiplas *máquinas virtuais* em um único computador, compartilhando, desta forma, os recursos desse equipamento entre os ambientes virtualizados. Uma máquina virtual (VM) é um software que simula a operação de um computador e que executa programas exatamente como se fosse a máquina real. Desta maneira, as máquinas virtuais permitem executar diferentes sistemas operacionais e, consequentemente, vários aplicativos diferentes, tudo sobre o mesmo hardware físico. Assim, a virtualização oferece uma maneira para extrair o máximo da capacidade computacional disponível.

De acordo com Popek e Goldberg (1974), cada máquina virtual deve atender a três requisitos básicos, que são:

- Eficiência: qualquer instrução de máquina, desde que não comprometa o funcionamento do sistema, deve ser executada diretamente sobre o hardware, sem a interferência da máquina virtual;
- Controle de recursos: uma máquina virtual deve ter controle total sobre os recursos virtualizados. É indesejável que qualquer programa executando sobre a VM acesse os recursos diretamente;
- Equivalência: um programa que executa sobre uma VM deve apresentar comportamento idêntico ao apresentado quando executado sobre uma máquina física equivalente. Neste caso, duas exceções são consideradas: eventualmente algumas instruções podem ter seu tempo de execução aumentado; e pode haver conflito de acesso a recursos, que devem ser resolvidos apropriadamente.

Existem diferentes maneiras de implementação de máquinas virtuais. Na próxima seção, são apresentados alguns conceitos e estruturas que são comuns a todas as técnicas, bem como, uma breve abordagem sobre a composição e o funcionamento de cada uma delas.

2.2 Técnicas para a Implementação de Máquinas Virtuais

A camada de software responsável pela interação direta com o hardware do computador é o sistema operacional (SO). Este denomina-se *hospedeiro* quando é executado sobre o hardware real, enquanto que os sistemas operacionais que são executados sobre as VMs são chamados de *hóspedes*.

O sistema operacional comunica-se com os componentes do computador através de interfaces, que são disponibilizadas pela microarquitetura do hardware. As arquiteturas modernas oferecem três tipos de interfaces, que são: as instruções de máquina privilegiadas, as instruções de máquina não-privilegiadas, as chamadas de sistema e as APIs (*Application Program Interfaces*). A Figura 1 (CARISSIMI, 2008) ilustra essas interfaces.

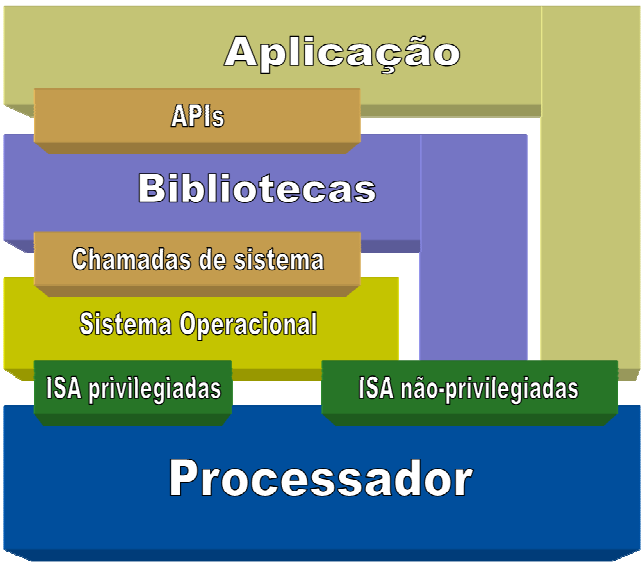

Figura 1. Interfaces genéricas de um sistema de computação (CARISSIMI, 2008)

As instruções de máquina privilegiadas (ou ISA privilegiadas) são acessíveis somente pelo sistema operacional, pois nesse conjunto encontram-se as instruções mais críticas do hardware, como por exemplo, as de manipulação de entrada e saída e de interrupções de CPU. Já as instruções de sistema não privilegiadas podem ser acessadas também pelas bibliotecas e APIs, bem como diretamente por processos de usuário. As APIs, por sua vez, são interfaces para chamadas de sistema (*system calls*) a serem utilizadas pelos processos de usuário de forma a padronizar e facilitar a manutenção destes. É comum que as chamadas de sistema sejam encapsuladas nessas interfaces.

Essa subdivisão em camadas garante que os recursos de hardware sejam acessados de forma controlada através do sistema operacional, que é a única camada que tem acesso às instruções privilegiadas (CARISSIMI, 2008). Além disso, torna-se possível que os SOs utilizem padrões pré-determinados para comunicação com as interfaces disponíveis e assim facilitando o acesso a elas.

Segundo Carissimi (2008), considerando essa estrutura e funcionamento dos sistemas operacionais modernos, é possível definir três técnicas de implementação de máquinas virtuais, que são: *emulação*, *máquina virtual de processo* e *máquina virtual de sistema*[1]. Nas próximas seções, apresentam-se as características gerais de cada uma delas.

2.2.1 Emulação

A *emulação* é a forma mais simples de se implementar o conceito de virtualização. Através desta técnica, o *emulador* (ou *runtime*) é o responsável por simular o funcionamento de um determinado conjunto de hardware (independentemente do hardware físico) que é oferecido ao sistema operacional hóspede. O emulador nada mais é do que um programa compatível com o SO hospedeiro, que o executa da mesma forma que qualquer outro aplicativo.

Ao utilizar a técnica de emulação, obtém-se a maior vantagem do uso de máquinas virtuais, que é a total portabilidade das VMs, ou seja, qualquer máquina virtual pode ser executada sobre qualquer hardware, pois o sistema hóspede não tem acesso a nenhum outro hardware exceto aquele oferecido pelo emulador. Por outro lado, tem-se como desvantagem a perda de performance, porque o emulador deve oferecer um conjunto de dispositivos emulados que seja possível de ser executado sobre praticamente qualquer hardware físico. Portanto, muitas vezes esses dispositivos são “genéricos” e menos eficientes.

A perda de desempenho também se deve ao fato de que toda e qualquer instrução executada pelo hóspede será enviada ao hardware “falso” (oferecido pelo emulador), que será o responsável por traduzi-las de forma que essas possam ser executadas pelo hardware real. Tal tarefa é dispendiosa, tornando a execução mais lenta (MADDEN, 2009). Como exemplos de softwares que utilizam técnicas de emulação, é possível citar os emuladores de sistemas operacionais antigos, como do Mac OS, Amiga, MSX e outros.

[1] Carissimi (2008) fornece tal classificação para os tipos de máquinas virtuais existentes no mercado, porém o próprio autor, em outras publicações, adota terminologia diferente, como, por exemplo, em CARISSIMI (2009).

2.2.2 Máquina Virtual de Processo

A segunda maneira, chamada de *máquina virtual de processo,* consiste em criar um aplicativo, que executa sobre um determinado sistema operacional "real" (i.e., instalado no hardware físico) e que oferece um ambiente de execução para outras aplicações. Esse ambiente pode possuir um conjunto de instruções meramente abstratas que são interpretadas, de forma a gerar as instruções de máquina que serão executadas na máquina real. Um exemplo de máquina virtual de processo que trabalha desta forma é a JVM – *Java Virtual Machine* (CARISSIMI, 2008).

Em uma máquina virtual de processo, também é possível que o aplicativo que executa sobre o hardware real ofereça emulação de chamadas de sistema de outros sistemas operacionais que são interpretadas e executadas sobre o sistema real. Esta é a forma de funcionamento de softwares como o *VMware Player*, que permite, por exemplo, executar o sistema operacional Linux sobre uma máquina real com o sistema operacional Windows. Outros produtos que fazem uso dessa técnica são: o *VirtualPC*, da Microsoft, e o projeto *open-source* QEMU. É importante salientar que, da mesma forma que os processos do sistema operacional, as máquinas virtuais de processo também são entidades efêmeras, que só existem durante a sua execução.

A Figura 2 (CARISSIMI, 2008) ilustra uma máquina virtual de processo.

Figura 2. Máquina virtual de processo e sua relação com os sistemas hóspede e hospedeiro (CARISSIMI, 2008)

2.2.3 Máquina Virtual de Sistema

Outra abordagem de implementação de máquinas virtuais é conhecida por *máquina virtual de sistema*, *hipervisor* ou *VMM (Virtual Machine Monitor).* Tal técnica consiste em inserir uma camada de software entre o hardware físico e o SO hospedeiro, com o objetivo de proteger o acesso deste último ao hardware. Essa camada de software oferece o mesmo conjunto de instruções do hardware real ou algum outro conjunto, que será então interpretado para execução no hardware.

Neste caso, é imprescindível que essa camada esteja disponível sempre que o computador estiver operando e acessível simultaneamente a vários programas (IBM, 2005). Exemplos de softwares de virtualização que utilizam esta técnica são o *VMware ESX Server*, o *Xen*, o *Microsoft Hyper-V* e algumas distribuições de Linux modificadas. Essas distribuições são conhecidas por *Kernel-based Virtual Machines* (KVM) e transformam o próprio sistema operacional em um hipervisor. A Figura 3 (CARISSIMI, 2008) esquematiza um hipervisor.

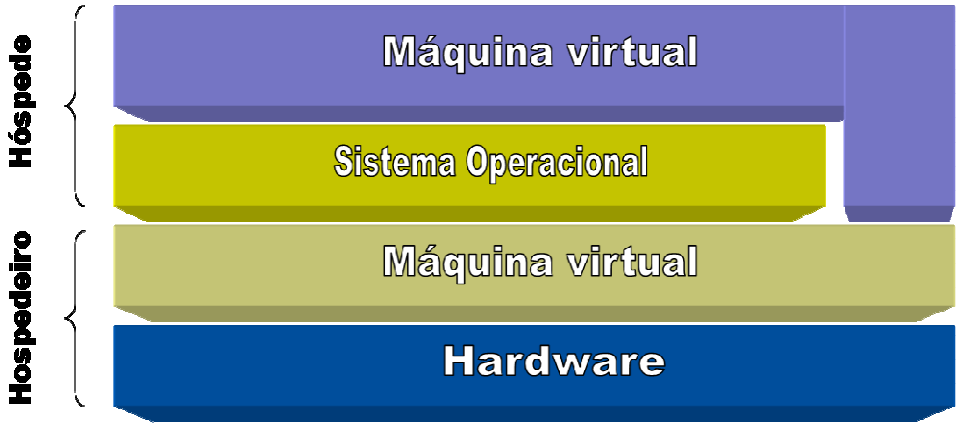

Figura 3. Hipervisor e relação com sistemas hóspede e hospedeiro (CARISSIMI, 2008)

Diferentemente das máquinas virtuais de processo, os hipervisores, por estarem em uma camada inferior à do sistema operacional e ligados diretamente ao hardware, perduram durante todo o tempo em que o computador estiver em utilização (CARISSIMI, 2008).

É possível classificar, ainda, as técnicas de implementação de máquinas virtuais de sistema em dois grupos, conforme a estratégia de virtualização que cada uma utiliza. Esses grupos são: virtualização total e paravirtualização (CARISSIMI, 2008). Ambos encontram-se descritos nas próximas seções.

2.2.3.1 Virtualização Total

Ainda nos anos 1960, a IBM apresentou o conceito de VMM no seu mainframe *System/360 Model 67*. Esse equipamento virtualizava todas as interfaces do hardware físico através de uma VMM. Com a capacidade de se executar sistemas operacionais sobre outros sistemas operacionais, cada máquina virtual podia então executar uma instância do seu próprio sistema operacional (JONES, 2006). Tal técnica de virtualização é conhecida por *virtualização total* e consiste em prover uma réplica fiel do hardware real às máquinas virtuais que forem executadas.

A virtualização total difere da emulação pelo fato de que a virtualização pode ser oferecida como um serviço do SO hospedeiro ou até mesmo como uma funcionalidade embutida no próprio hardware. Já a emulação depende do hospedeiro para funcionar, pois o emulador é executado da mesma forma que qualquer outro aplicativo. Além disso, os emuladores oferecem ao SO hóspede apenas conjuntos "genéricos" de dispositivos emulados, enquanto que a virtualização total permite, ainda, fornecer acesso a alguns recursos do hardware físico – normalmente a itens secundários, como placas de som e outros dispositivos não-essenciais.

Apesar da simplicidade para criação e manutenção de máquinas virtuais, esta técnica apresenta algumas desvantagens. Por exemplo, devido à enorme diversidade de dispositivos no mercado, torna-se muito difícil programar uma VM que simule o comportamento exato de cada um deles. Normalmente o hipervisor oferece suporte a um pequeno conjunto de dispositivos essenciais, como teclado, mouse, placas de vídeo e de rede, interfaces de disco e outros. Essa solução quase sempre causa a subutilização dos recursos de hardware disponíveis, pois os dispositivos oferecidos pela ferramenta são "genéricos", ou seja, dificilmente são idênticos aos recursos do hardware físico. Também existem perdas de desempenho na execução das VMs, pelo fato de que o hipervisor deve tratar eventuais *traps*[2] (exceções) no hardware real, entre outros problemas.

Alguns softwares que trabalham desta maneira são os programas *VirtualBox*, da Sun, *VMware Server*, *VMware Workstation* e *VMware Fusion*. Este último é voltado aos usuários do sistema Mac OS X, da Apple. A Figura 4 (CARISSIMI, 2008) exibe um esquema de um hipervisor de virtualização total.

[2] Em sistemas operacionais, uma *trap*, ou exceção, é uma interrupção na execução do sistema, normalmente causada por alguma condição excepcional em um processo de usuário, como uma divisão por zero ou acesso inválido à memória.

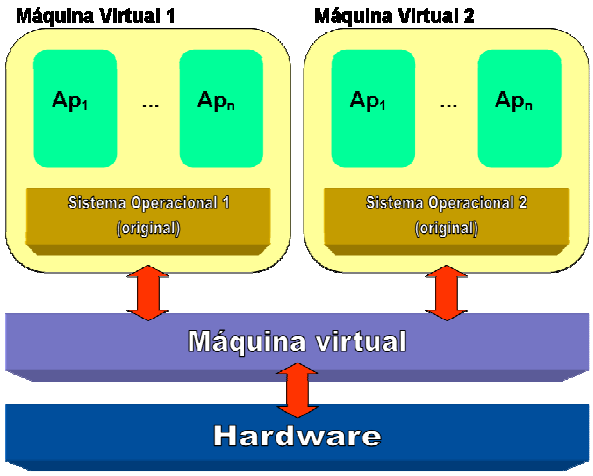

Figura 4. Virtualização total (CARISSIMI, 2008)

2.2.3.2 Paravirtualização

Outra técnica para o desenvolvimento de hipervisores é chamada de *paravirtualização*, na qual o sistema hóspede é modificado de forma que a VMM seja incumbida do tratamento de eventuais casos de chamadas ou instruções consideradas "sensíveis", isto é, que possam causar conflitos ou *traps* no hardware real. Assim, o tratamento das chamadas ou instruções "não-sensíveis" deixa de ser necessário, o que significa um ganho em desempenho. Além disso, nessa técnica os dispositivos de hardware são acessados diretamente pelo SO hóspede, através de *drivers* fornecidos pela própria VMM. A Figura 5 (CARISSIMI, 2008) demonstra esquematicamente o funcionamento de um hipervisor com paravirtualização.

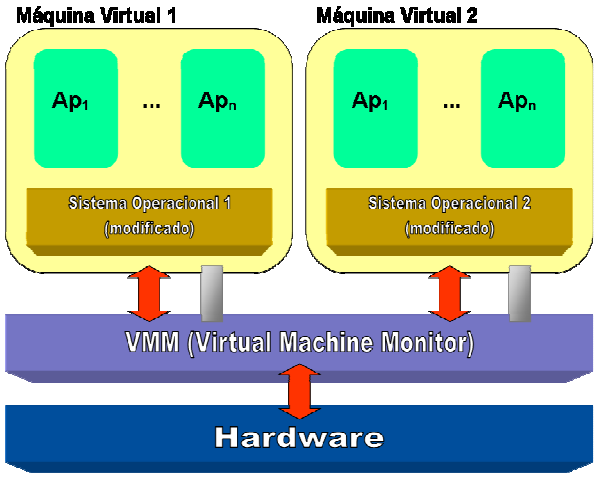

Figura 5. Paravirtualização (CARISSIMI, 2008)

Apesar da vantagem apresentada por essa técnica em desempenho e utilização de recursos do hardware físico, existe a necessidade de modificar o sistema operacional a ser executado como hóspede. A paravirtualização é aplicada no software Xen, que oferece uma versão modificada do Linux chamada de XenLinux. As mudanças efetuadas nessa distribuição, que só executa sobre o hipervisor Xen, estão fortemente ligadas ao *kernel* do sistema, de maneira que algumas modificações sobre o hipervisor exigem até mesmo a recompilação do *kernel*.

Em 2005, a VMware apresentou outra estratégia de desenvolvimento da tecnologia de paravirtualização. Naquele ano, a empresa propôs uma interface de virtualização chamada de *Virtual Machine Interface* (VMI), como um mecanismo padronizado de comunicação entre o SO hóspede e o hipervisor. Basicamente, essa interface consiste em outra camada adicional de software entre o hardware físico e os sistemas paravirtualizados. No ano de 2006, a empresa apresentou a especificação da VMI, de código aberto, disponível para *download* no *site* da empresa, sujeito à licença GPL 2.0[3].

Assim, o sistema operacional hóspede pode ser executado tanto nativamente sobre o hardware físico quanto virtualizado sobre qualquer VMM compatível com o padrão. Isso é alcançado devido a todas as chamadas à VMI possuírem duas implementações: instruções nativas para o hardware físico e chamadas indiretas à camada de virtualização presente entre o sistema hóspede e o hipervisor. A Figura 6 (VMWARE, 2009a) ilustra o funcionamento da VMI.

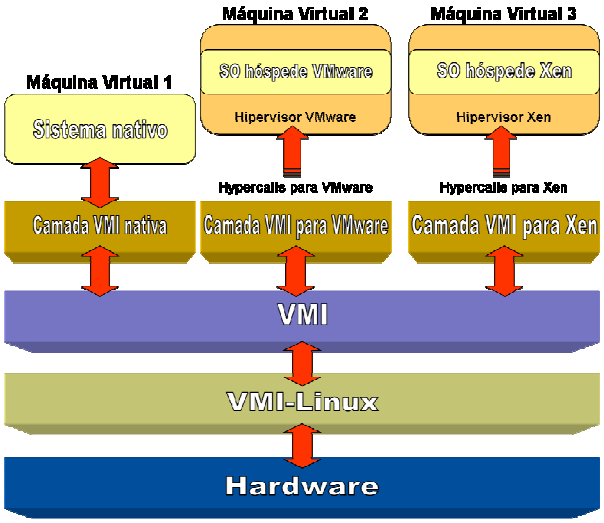

Figura 6. Paravirtualização pela Virtual Machine Interface (VMI) (VMWARE, 2009a)

[3] Ver FREE SOFTWARE FOUNDATION, 1991.

2.3 Ferramentas para Virtualização

Atualmente, existe uma grande quantidade de softwares que implementam os conceitos de virtualização, cada qual com suas características técnicas. Nesta seção, serão descritos os softwares mais conhecidos e utilizados de virtualização.

2.3.1 Sun VirtualBox

A primeira versão do software *VirtualBox* foi lançada em 15 de janeiro de 2007 pela empresa alemã *Innotek*, que a oferecia ao mercado tanto através de uma licença comercial, como também por uma versão gratuita sujeita à licença de uso *VirtualBox Personal Use and Evaluation License (PUEL*). A licença *VirtualBox PUEL* autorizava a utilização do pacote completo do software *VirtualBox* e de todos os seus recursos, sem custos para atividades domésticas, acadêmicas e para testes, porém não permitia a redistribuição do software (SUN, 2009a). Estava disponível, ainda, uma versão de código aberto, chamada *VirtualBox OSE* (*Open-Source Edition*), sujeita às regras da *General Public License* (GPL), versão 2.0, da GNU. Nesta modalidade, o software *VirtualBox* era entregue como software gratuito, inclusive com o código-fonte disponível (FREE SOFTWARE FOUNDATION, 1991), porém alguns recursos avançados não estavam disponíveis (SUN, 2009a).

Em fevereiro de 2008, a *Sun Microsystems* adquiriu a *Innotek*, passando a coordenar o desenvolvimento do *VirtualBox* e a integrá-lo à sua plataforma de virtualização *xVM* (SUN, 2009a). Mesmo após a aquisição da Innotek, a Sun continua a oferecer o *VirtualBox* através das mesmas licenças de uso que eram adotadas pela empresa alemã.

Algumas das principais características do *VirtualBox* incluem o suporte ao *Open Virtualization Format* (OVF), que permite a interoperação com outras plataformas de virtualização compatíveis com esse padrão. Além disso, o *VirtualBox* foi construído sobre uma arquitetura aberta, que oferece acesso às suas APIs através de serviços locais ou interfaces de *web services.* Dessa forma, o software possibilita a customização de máquinas virtuais para a distribuição.

O *VirtualBox*, que implementa a técnica de máquina virtual de sistema, utiliza instruções SMP[4], suporta até 32 CPUs virtuais e é compatível com processadores que ofereçam extensões de virtualização (Intel VT-x e AMD-V). Além disso, o software também oferece suporte nativo a gráficos *Direct3D* e *OpenGL*, permitindo que as máquinas virtuais

[4] SMP (*symmetric multiprocessing*, ou multiprocessamento simétrico), técnica utilizada nas arquiteturas de microprocessadores mais modernos, que permite que dois ou mais processadores idênticos possam conectar-se a um único banco de memória principal compartilhado entre eles.

realizem operações de vídeo diretamente através do *driver* da placa de vídeo do hardware físico e, consequentemente, com um maior desempenho.

Ainda pode-se citar o fato de que o *VirtualBox* permite a criação de múltiplos *snapshots*[5] para cada máquina virtual, oferecendo uma maior segurança no que se refere aos *backups* das máquinas virtuais. Além disso, tendo à disposição vários *snapshots*, o usuário pode, por exemplo, voltar a máquina virtual a diversos estados anteriores, não somente ao último estado salvo.

Em relação à compatibilidade com sistemas operacionais hospedeiros, o *VirtualBox* permite a utilização do *Sun Solaris OS* (32 e 64 bits: *OpenSolaris* ou *Solaris 10 OS 5/08* e superiores), Linux (32 e 64 bits: distribuições *Debian*, *Fedora*, *Gentoo*, *Red Hat Enterprise*, *SUSE*, *Ubuntu* e *Mandriva*), MS Windows (32 e 64 bits: XP, *Server 2003*, *Server 2008* e *Vista*) e Mac OS X (somente Intel: qualquer versão). Já quanto aos SOs hóspedes, o software dá suporte a praticamente todos os sistemas operacionais compatíveis com a plataforma x86 (SUN, 2009a).

2.3.2 Xen

O hipervisor Xen surgiu em um projeto de pesquisa, coordenado por Ian Pratt na Universidade de Cambridge, em 2003. Ian Pratt, mais tarde, tornou-se o fundador da XenSource, Inc., que é a entidade que dá suporte ao desenvolvimento do projeto de código aberto Xen e que também comercializa versões empresariais do software. Ainda em 2003, foi disponibilizada a primeira versão pública do Xen.

Em outubro de 2007, a XenSource, Inc. foi adquirida pela empresa *Citrix Systems*. A partir de então, os produtos da XenSource foram rebatizados com a marca da nova proprietária. Essas mudanças foram acompanhadas pelo surgimento do *Xen Project Advisory Board* (Xen AB), o qual possui entre seus membros a própria Citrix, além de empresas como IBM, Intel, HP, Novell, Oracle/Sun, entre outras. O Xen AB tem a função de conselho consultivo em relação às atividades de desenvolvimento do projeto Xen, bem como de gerenciar a marca comercial “Xen” (CITRIX, 2009a).

[5] Os *snapshots* são conjuntos de arquivos que guardam uma “imagem” fiel da máquina virtual em um dado ponto no tempo. Sua função é facilitar a restauração da máquina virtual em caso de problemas ou para recuperação de *backups.* Entre esses arquivos, encontram-se todo o conteúdo do disco rígido virtual, o conteúdo da memória RAM da máquina virtual e demais arquivos necessários para o funcionamento do SO hóspede, exatamente como estavam no momento da criação do *snapshot*.

O Xen é totalmente baseado em código aberto, sob a licença GPL 2.0, da GNU, estando disponível, sem custo, tanto para cópia de seu código-fonte quanto para cópia de sua versão pronta para uso (FREE SOFTWARE FOUNDATION, 1991). Atualmente, cerca de 20 empresas e toda a comunidade de desenvolvedores Xen trabalham no desenvolvimento colaborativo do software (CITRIX, 2009a).

O Xen é um exemplo de máquina virtual de sistema desenvolvido com técnicas de paravirtualização. Atualmente, o software suporta processadores das arquiteturas Intel x86 e compatíveis, x86-64 e compatíveis, Intel IA-64 e PowerPC (inclusive de gerações mais antigas de PowerPC, permitindo o seu uso também em equipamentos tecnicamente defasados). Além disso, também possui suporte para processadores com instruções SMP.

O software Xen suporta computadores com até 32 processadores físicos e permite que cada VM seja configurada com até 32 processadores (em equipamentos de arquitetura IA64, este número aumenta para 64 CPUs por máquina virtual). Como sistemas hospedeiros, é possível utilizar o Linux com *kernel* versão 2.4 ou 2.6, em distribuições como a NetBSD, FreeBSD e outras.

No *site* do fabricante, existem dois pacotes do Xen disponíveis para *download.* O primeiro deles oferece o Xen separadamente, que pode ser instalado e configurado em ambientes Linux compatíveis. Já o segundo pacote possui o Xen e também uma distribuição chamada de *XenLinux*, que contém uma versão modificada do *kernel* versão 2.6 desse sistema operacional. As modificações no núcleo do Linux que são promovidas pela configuração do Xen têm o objetivo de fazer com que o sistema operacional hospedeiro ofereça às máquinas virtuais uma microarquitetura virtualizada própria, chamada de *Xen/x86*, que é muito semelhante à arquitetura x86 convencional. Porém, na arquitetura *Xen/x86*, as instruções privilegiadas foram substituídas por chamadas (*hypercalls*) ao hipervisor Xen. Assim, torna-se possível que as chamadas às instruções “sensíveis”, que podem causar *traps* no SO hospedeiro, sejam controladas pelo hipervisor.

Já para sistemas hóspedes, é possível executar várias distribuições de Linux e também o Microsoft Windows XP e o OpenSolaris. Todavia, o Xen exige algumas alterações nos códigos-fonte do *kernel* desses sistemas para tornar possível a execução em máquinas virtuais. Essas alterações permitem que os sistemas hóspedes sejam capazes de interagir com a arquitetura Xen/x86 oferecida pelo hipervisor e suas particularidades (PRATT et. al., 2004).

Como características, pode-se destacar que o Xen fornece controle e priorização de acesso aos recursos, permitindo a configuração de prioridades e limites para o uso de

processador, memória, disco e tráfego de rede. Além disso, o software também suporta migração “quente” de máquinas virtuais (*live migration*). Esta opção possibilita que uma máquina virtual que está sendo executada em um determinado hardware físico seja transferida através da rede para outro equipamento. Durante essa transferência, a máquina virtual fica em um estado “suspenso” (sem efetuar processamento durante alguns instantes) até que o equipamento de destino já tenha recebido os dados necessários para retomar a execução da máquina virtual. O recurso de *live migration* do Xen é de grande utilidade em estruturas de virtualização de maior porte, pois possibilita o balanceamento da carga entre os equipamentos físicos da rede, de maneira a otimizar a utilização dos recursos de hardware disponíveis (CITRIX, 2008).

O uso da técnica de paravirtualização permite ao Xen oferecer máquinas virtuais que são executadas com um desempenho muito próximo ao de um sistema Linux tradicional. Pratt et. al. (2004) efetuaram testes de *benchmark*[6] sobre um computador com Linux sem virtualização, depois em uma máquina virtual sobre o Xen e finalmente em uma máquina virtual sobre o VMware Workstation, visando determinar a perda de desempenho devido ao uso de cada ferramenta de virtualização.

Resumidamente, durante esses testes foram aplicados quatro métodos de *benchmark* bastante difundidos no mercado. O primeiro, chamado CINT2000, é parte integrante da suíte SPEC CPU2000 e consiste em doze operações aritméticas com números inteiros (SPEC, 2009). O segundo método é efetuar uma compilação do *kernel* de uma distribuição Linux, para simular a carga de trabalho que é executada por um computador em tarefas típicas de desenvolvimento de software (KAKULAVARAPU, 2008). O terceiro teste, chamado OSDB[7] OLTP[8], visa medir o desempenho de acesso a banco de dados, ou seja, no caso deste *benchmark* existe um intenso acesso ao(s) disco(s) rígido(s) da máquina (OLIVEIRA, 2007). Por fim, o último teste, chamado SPECweb99, tem por objetivo determinar o desempenho do computador quando utilizado como um servidor *web*. Nesses testes são simuladas múltiplas conexões simultâneas, acesso a arquivos e outras atividades comuns para um equipamento desta categoria (SPEC, 2009). A Figura 7 (PRATT et.al., 2004) mostra os resultados obtidos nesses testes.

---

[6] *Benchmarks* são procedimentos para testes de desempenho computacional específico de um determinado aspecto do equipamento. Existem *benchmarks* especializados em simular condições de uso extremo de processador, memória RAM, acesso a disco e tráfego de rede, entre outros.

[7] *Open Source Database Benchmark*, um pacote para testes de desempenho de bancos de dados.

[8] *Online Transaction Processing*, ou Processamento de Transações em Tempo Real, são sistemas que registram todas as transações ocorridas durante uma determinada operação.

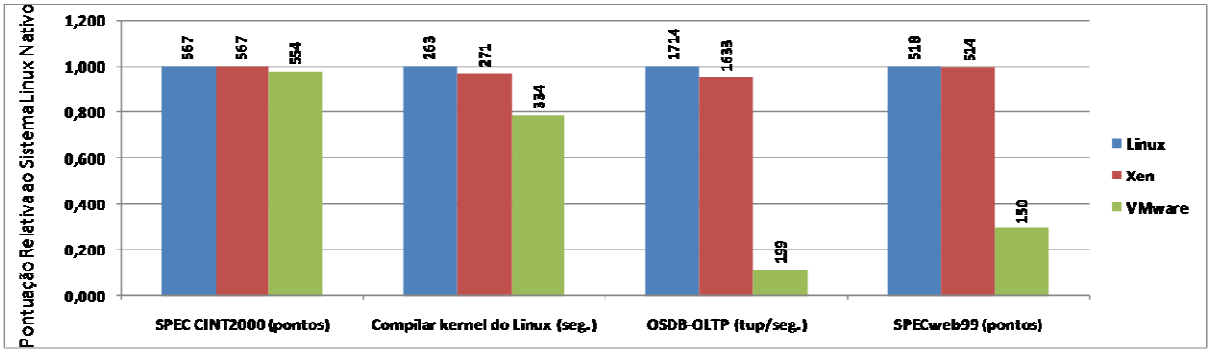

Figura 7. Resultados da execução de dois *benchmarks* em máquinas virtuais e em uma máquina Linux nativa (PRATT et. al., 2004)

2.3.3 VMware Server

A empresa norte-americana VMware, sediada em Palo Alto, Califórnia, foi fundada em 1998 e hoje é considerada uma das líderes do mercado mundial de virtualização. Atualmente, a companhia detém um portfólio de produtos que possui soluções para diversas aplicações dessa tecnologia e oferece desde softwares apenas para a execução de máquinas virtuais até sistemas mais complexos, com recursos de migração de máquinas virtuais, gerenciamento dos recursos de hardware físico, *snapshots* e vários outros recursos.

A VMware oferece softwares gratuitos e comerciais, sendo que sua lista de produtos é dividida em três grandes grupos:

- Utilitários diversos: neste grupo, a empresa oferece o *VMmark*, que é uma ferramenta para *benchmark* de máquinas virtuais, além do *VMware Studio* – uma suíte para criação e gerenciamento de pacotes compatíveis com máquinas virtuais no padrão OVF (*Open Virtualization* Format) – e do *VMware Player*, programa gratuito que permite somente a execução de máquinas virtuais (no conceito de *runtime*), sem a necessidade de instalação de todo o software de virtualização na máquina.

- Produtos para *desktops*: nesta categoria estão os programas voltados à virtualização de estações de trabalho, entre eles o *VMware Workstation* e o *VMware Fusion*. Ambos oferecem a possibilidade de executar uma ou mais máquinas virtuais sobre um sistema hospedeiro, na configuração de máquinas virtuais de processo. O *Workstation* é um produto voltado à execução em ambientes Windows e Linux, enquanto que o *Fusion* é direcionado ao mercado de Macintosh.

- Produtos para servidores e *datacenters*: agrega a maior parte dos softwares oferecidos pela fabricante, em um total de 17 produtos, cada um com suas características. Nesse grupo, além de soluções comerciais e ferramentas para gerenciamento de máquinas virtuais, encontram-se o VMware ESXi e o VMware Server, que são oferecidos gratuitamente para *download*.

Neste trabalho, optou-se pelo uso do VMware Server. Apesar do fato de o VMware ESXi oferecer maior número de recursos avançados do que o VMware Server, o ESXi necessita de combinações de hardware específicas para seu funcionamento, tais como, combinação de processador, *chipset* da placa-mãe e controladora de discos, além da disponibilidade de sistemas de *storage* compatíveis. Esses requisitos impossibilitaram os testes com esse software.

Por outro lado, o VMware Server, por implementar a técnica de máquina virtual de sistema, é facilmente configurável para utilização em praticamente qualquer equipamento que atenda aos requisitos mínimos de hardware. A Tabela 1 demonstra algumas diferenças entre o VMware ESXi e o VMware Server.

Tabela 1. Comparativo entre as soluções VMware gratuitas para servidores (VMWARE, 2009a)

	VMware Server	**VMware ESXi**
Arquitetura	Máquina virtual de sistema	Hipervisor (virtualização total)
Sistema operacional hospedeiro	Windows ou Linux	Nenhum
Caso de uso típico	Teste/desenvolvimento	Produção
Necessita servidor dedicado	Não	Sim
Opção de gerenciamento centralizado	Não	Sim

A versão atual do VMware Server é a 2.0.1, disponível gratuitamente tanto para sistemas hospedeiros de 32 bits quanto para sistemas de 64 bits. O VMware Server provê recursos como a virtualização de processadores com instruções SMP e criação de *snapshots*. Esse software pode gerenciar até 16 processadores por computador físico, com um máximo de 4 VMs sendo executadas concorrentemente sobre cada processador, além

de até 8 GB de memória RAM para cada máquina virtual. Como sistema operacional existe suporte ao *Microsoft Windows Server 2003* ou *2008* (exceto o *2008 Server Core*) ou algumas distribuições Linux para servidores, tais como *Mandriva Corporate Server, Red Hat Enterprise, SUSE Enterprise Linux* e *Ubuntu Server*.

Como sistemas operacionais hóspedes, o VMware Server fornece suporte ao Microsoft Windows nas versões *2000 Server*, *XP Professional*, *Vista Business* e *Ultimate*, *2003 Server* (*Small Business, Standard, Web* e *Enterprise*) e *2008 Server* (*Standard* e *Enterprise*), bem como as distribuições Linux *Red Hat*, *Mandake*, *Mandriva*, *SUSE*, *openSUSE*, *Open Enterprise Server* (*OES*) e *Ubuntu*, além de versões do Sun Solaris e do Novell NetWare.

Destaca-se que o VMware Server não oferece suporte à migração "quente" de máquinas virtuais (*live migration*). Tal opção é oferecida somente em produtos como o *VMware vSphere* e *VMware ESX*, ainda assim como um recurso que deve ser adquirido separadamente, chamado *VMotion*. Outros recursos comuns em ferramentas de virtualização, como por exemplo o acesso das VMs a pastas compartilhadas (*shared folders*) também não estão disponíveis no VMware Server (VMWARE, 2009a).

2.4 Considerações Finais

Nesta seção, é apresentada uma breve síntese das características das ferramentas descritas neste trabalho. A Tabela 2 resume os principais aspectos de cada uma dessas ferramentas, de maneira a facilitar a comparação entre elas.

Tabela 2. Principais características técnicas das ferramentas estudadas

	Sun VirtualBox	**Xen**	**VMware Server**
Arquitetura do hospedeiro	Máquina virtual de sistema (virtualiza total)	Hipervisor (paravirtualização)	Máquina virtual de sistema (virtualização total)
Arquiteturas hóspedes compatíveis	Intel X86, X64 e compatíveis	Intel X86, X64, IA-64 e compatíveis, e PowerPC	Intel X86, X64 e compatíveis
Compatível com instruções Intel VT-X e AMD-V	Sim	Sim	Sim

Número máximo de CPUs virtuais	32	32	64
Permite *live migation* de VMs	Não	Sim	Não[9]
Permite a criação de múltiplos snapshots	Sim	Sim	Sim
SOs hospedeiros compatíveis	Sun Solaris OS; Versões do Windows a partir do Windows XP; Linux (7 distribuições) Mac OS X (Intel)	Linux (*kernel* 2.4 ou 2.6); XenLinux	Versões do Windows a partir do Windows 2000 Server; Linux (6 distribuições)
SOs hóspedes compatíveis	Praticamente qualquer SO compatível com x86	Linux (mais de 20 distribuições); Versões do Windows a partir do Windows XP; OpenSolaris[10]	Linux (8 distribuições); Versões do Windows a partir do Windows XP; Sun Solaris OS; Novell NetWare
Preço e licença de uso	Gratuito (GNU GPL 2.0)	Gratuito (GNU GPL 2.0)	Gratuito (licença proprietária)
Fabricante e website	Sun (www.virtualbox.org)	Citrix (www.xen.org)	VMware (www.vmware.com)

Neste trabalho, procurou-se dar preferência a softwares de gratuitos. Seguindo esse critério, foram selecionados para testes os programas Sun VirtualBox, VMware Server e Citrix XenServer, sendo que este último foi desenvolvido sobre a estrutura do Xen.

Para avaliação da migração “quente” de máquinas virtuais entre servidores físicos, foi selecionado somente o Citrix XenServer, devido ao fato de que somente este software oferece tal recurso em sua versão gratuita.

[9] A VMware oferece o recurso de migração de máquinas virtuais como um produto comercial separado, chamado *VMotion*, que só está disponível em algumas versões da suíte paga *vSphere*.

[10] Devido às características da técnica de paravirtualização, os sistemas hóspedes a serem executados com o Xen precisam sofrer algumas modificações que viabilizem sua execução.

3 Métodos para *Benchmark*

Há até alguns anos atrás, uma forma simples de se determinar qual computador oferecia o melhor desempenho era verificar o *clock* do seu processador. Quanto maior o *clock*, maior a performance do equipamento. Até então, detalhes técnicos como o número de *cores* do processador, a velocidade do barramento de memória RAM, etc. não influenciavam no tempo necessário para a execução das tarefas cotidianas (TORRES, 2009).

Com a evolução das arquiteturas de computadores, somente o *clock* do processador não é mais suficiente na tarefa de escolher o melhor equipamento. Por isso, foram criadas metodologias de teste (*benchmarks*) que podem ser executadas em diferentes configurações de sistemas, permitindo comparações entre os resultados obtidos. Há alguns anos, tais testes permitiram demonstrar, por exemplo, que processadores Pentium 4 nem sempre eram mais eficientes que processadores Athlon XP, mesmo que o produto da Intel operasse a frequências de *clock* mais altas que as utilizadas pelo *chip* da AMD (TORRES, 2009).

Um *benchmark* consiste, de maneira simplificada, em simular uma determinada carga computacional (*workload*) sobre sistemas diferentes, para que a performance e o custo de cada sistema sejam medidos e registrados. A *performance* normalmente é medida em “quantidade de trabalho por segundo”, enquanto o *custo* é uma métrica baseada em cinco anos de TCO[11] (*total cost of ownership*) do equipamento avaliado. Esses dois fatores em conjunto fornecem uma relação de custo/performance.

Basicamente, é possível definir dois tipos de testes de *benchmark*, os chamados *benchmarks* “genéricos” e *benchmarks* “sintéticos”, também conhecidos por *benchmarks* de “domínio específico”. Os *benchmarks* genéricos são os mais comuns e servem como uma estimativa generalizada de desempenho. Tais testes consistem em executar *workloads* diversificados sobre os equipamentos, ou seja, são processadas cargas de trabalho que simulam o cotidiano de um computador típico, tais como a execução de processadores de texto, planilhas eletrônicas, jogos e outros tipos de software. Alguns desses *benchmarks* já se popularizaram de tal maneira que existem fabricantes que chegam a apresentar o desempenho de seus produtos utilizando os resultados obtidos nesses testes.

[11] As modelagens de *total cost of ownership*, ou custo total de propriedade, contabilizam sistematicamente todos os custos, diretos e indiretos, relacionados a um investimento em TI durante seu ciclo de vida, inclusive os valores de aquisição, contratos, operação e manutenção, bem como da gestão do final de sua vida útil (FEC, 2007).

Porém, algumas vezes os *benchmarks* genéricos podem se mostrar ineficientes, haja vista que o comportamento dos sistemas pode variar muito dependendo do tipo de aplicação que é executada sobre eles. Isso ocorre pois normalmente um sistema é projetado tendo em vista um conjunto específico de atividades, de forma que possam existir tarefas que não sejam nem mesmo possíveis de serem executadas nesse sistema. Por esse motivo, tais testes não oferecem as diretrizes necessárias para seleção de um sistema que irá trabalhar, por exemplo, como um servidor de banco de dados ou de aplicações, pois nesses ambientes, a performance do sistema será determinada mais pelos algoritmos utilizados por essas aplicações do que unicamente pelo desempenho atingido pelos componentes do sistema (GRAY, 1993).

Por sua vez, os *benchmarks* sintéticos foram criados para preencher tal lacuna deixada pelos testes genéricos. Esses *benchmarks* visam simular a resposta do sistema ou componente testado através da execução de programas desenvolvidos especialmente para impor o tipo de comportamento desejado (KS, 1996). Cada teste desta categoria utiliza-se de uma carga computacional "sintética", que simula o processamento de aplicações típicas do domínio desejado. O desempenho alcançado ao processar tal *workload* em sistemas computacionais diferentes pode oferecer uma aproximação de seu comportamento dentro desse domínio de problemas (GRAY, 1993).

3.1 Pacotes para *Benchmark*

Atualmente, existe uma vasta gama de softwares para *benchmark* disponíveis. Os programas mais conhecidos são oferecidos comercialmente pelos seus desenvolvedores e alguns deles são considerados até mesmo um padrão industrial, sofrendo auditorias e verificações regulares. Também existe uma quantidade significativa de softwares para *benchmark* gratuitos e *open-source*.

Das entidades que oferecem programas comerciais para *benchmark*, pode-se destacar a SPEC (*Standard Performance Evaluation Corporation*), o TPC (*Transaction Processing Performance Council*) e a BAPCo (*Business Applications Performance Corporation*). A maioria dos programas oferecidos por essas organizações é amplamente difundida e fornece resultados aceitos cientificamente. Entre os programas gratuitos para *benchmark* destacam-se o *LINPACK*, *LAPACK*, *HPL*, *Dhrystone*, *Whetstone*, *Fhourstones*, *Iometer*, *Coremark, HINT, Netperf, NetSpec* e diversos outros.

A SPEC oferece várias suítes de *benchmark*, todas sujeitas a licenças de uso proprietárias. Já os *benchmarks* oferecidos pelo TPC devem ser encomendados pelos fabricantes do hardware a ser testado, pois são voltados à simulação do *workload* típico de empresas que utilizam softwares de ERP. Por sua vez, a BAPCo – um consórcio sem fins lucrativos formado por mais de vinte empresas, entre elas AMD, Apple, Dell, Hewlett-Packard, Intel, Microsoft e Sony – fornece várias suítes com foco em testes de desempenho de computadores comuns, baseados no comportamento de sistemas operacionais e aplicativos populares, como processadores de texto e planilhas eletrônicas. Nesta seção, serão descritas algumas das ferramentas mais conhecidas e utilizadas para *benchmark*.

3.1.1 SPEC CPU2006

A suíte SPEC CPU2006 é composta pelos testes CINT2006 e CFP2006. O CINT2006, também conhecido por “SPECint”, executa 12 testes que efetuam cálculos aritméticos com números inteiros. Esses testes procuram simular o *workload* de programas como compiladores, interpretadores, processadores, jogos e outros. Já o CFP2006 (ou “SPECfp”) é formado por 17 testes que executam cálculos de ponto flutuante, imitando o comportamento de programas de simulação física, renderização de gráficos em 3D e processamento de imagens, entre outros (SPEC, 2009).

O SPEC apresenta como resultado um escore numérico, referente ao desempenho atingido pela combinação entre processador, memória e compilador utilizado. O pacote SPEC CPU2006 é oferecido como um produto comercial e sua licença custa entre US$ 200,00 (para entidades sem fins lucrativos e/ou educacionais) e US$ 800,00 (para os demais usuários).

3.1.2 LINPACK, LAPACK e HPL

O LINPACK é uma coleção de subrotinas escritas na linguagem Fortran, projetadas para a análise e resolução de problemas comuns de álgebra linear. Essa biblioteca foi desenvolvida para utilização nos supercomputadores dos anos 1970 e início dos anos 1980 e foi substituída pelo LAPACK (*Linear Algebra PACKage*), que tinha como objetivo fazer com que os pacotes EISPACK e LINPACK fossem executados com melhor performance em sistemas de alto desempenho atuais (NETLIB, 2009).

Apesar do LINPACK ter sido desenvolvido ainda na década de 70, ainda hoje esse *benchmark* é muito popular entre as comunidades de testes de desempenho computacional, sendo que uma variante do LINPACK chamada de HPL (*High Performance LINPACK)* é utilizada para a classificação dos 500 sistemas computacionais mais rápidos do mundo, no projeto TOP500 (TOP500.ORG, 2009).

Como resultado dos testes, o LINPACK fornece o tempo, em segundos, que o sistema necessita para terminar o processamento. Assim, esse *benchmark* permite comparações entre os tempos obtidos em diferentes equipamentos. As bibliotecas LINPACK, LAPACK e HPL estão disponíveis para *download* gratuito no site *www.netlib.org*.

3.1.3 Dhrystone

O *benchmark* Dhrystone foi publicado pela primeira vez em 1984 por Reinhold Weicker, da Siemens AG, e fora criado com a linguagem Ada (WEICKER, 1984). Mais tarde, foi publicada uma versão escrita na linguagem C, que se tornaria um dos testes mais utilizados pela indústria de microprocessadores. Atualmente, diversas versões do Dhrystone estão disponíveis para *download*, o que torna indispensável que em qualquer teste seja feito referências à versão utilizada do algoritmo e, principalmente, se o mesmo foi otimizado para alguma plataforma ou sistema operacional.

Todo o código do Dhrystone é composto por cálculos inteiros, operações com *strings*, operações lógicas e acessos à memória, de maneira a refletir as atividades de CPU que são executadas pela maioria dos aplicativos de uso geral. O resultado do *benchmark* Dhrystone é determinado através do tempo necessário para a execução de várias iterações de um único *loop*, que contém uma sequência fixa de instruções.

Ao fazer referências a testes com o Dhrystone, normalmente os resultados são medidos em “DMIPS”, ou “Dhrystone MIPS/MHz”. Tal medida representa o escore do Dhrystone dividido por 1.757, que é a pontuação que foi atingida pelo sistema VAX 11/780, que consiste em um equipamento que executa 1 MIPS (milhões de instruções por segundo) (YORK, 2002).

3.1.4 STREAM

O *STREAM* é uma ferramenta de *benchmarks* sintéticos, escrita originalmente na linguagem Fortran 77 e posteriormente disponibilizada também na linguagem C. Esse

software efetua medições de performance sobre quatro operações longas com vetores. Tais operações – *copy*, *scale*, *sum* e *triad* – são as mais comuns nesse tipo de atividade.

Através dessa metodologia, o *STREAM* permite determinar a largura de banda sustentável entre a memória e a FPU (unidade de ponto flutuante) do computador. Ao final dos testes, a ferramenta exibe a taxa de transferência de dados atingida pelo sistema, bem como os tempos médio, mínimo e máximo de cada operação que foi efetuada (MCCALPIN, 2009). O código-fonte do STREAM está disponível para *download* através do *site http://www.cs.virginia.edu/stream.*

3.1.5 Iometer

O *Iometer* é uma ferramenta de medição de desempenho de subsistemas de I/O, destinada tanto a sistemas comuns quanto a sistemas distribuídos. Originalmente, o pacote foi desenvolvido pela Intel, que o anunciou oficialmente em um evento da empresa em fevereiro de 1998. Desde então, o *Iometer* difundiu-se em grande escala na indústria.

Em 2001, a Intel descontinuou seu investimento no projeto Iometer, que então foi entregue à responsabilidade do *Open Source Development Labs* (OSDL). Em seguida, essa entidade registrou o projeto no site *SourceForge.net* e disponibilizou uma versão inicial do mesmo. Desde seu relançamento em 2003, o projeto Iometer é coordenado por um grupo de usuários, que tem mantido os trabalhos de desenvolvimento do software.

O Iometer é composto por dois programas, chamados Iometer e Dynamo. O Iometer é o centro de controle da ferramenta, que deve ser executado em uma única máquina com sistema operacional Microsoft Windows. Através da sua interface gráfica, é possível configurar o *workload* a ser executado e os parâmetros de operação, além de iniciar e parar os testes. Ao comando do Iometer, o Dynamo, que é o gerador de *workloads*, executa operações de I/O e grava registros de desempenho dessas operações, retornando-os ao Iometer ao final dos testes.

O pacote básico da ferramenta (os executáveis Iometer e Dynamo) é distribuído sob os termos da licença *Intel Open Source License*. Já outros módulos desenvolvidos mais recentemente são distribuídos através dos termos da licença pública GPL 2.0 da GNU (IOMETER, 2009). O *benchmark* Iometer pode ser obtido através do site *http://www.iometer.org*.

3.1.6 Bonnie e Bonnie++

O *benchmark* Bonnie++ foi desenvolvido por Russell Coker no início da década de 2000, baseado no software Bonnie, que foi criado em 1988 por Tim Bray, utilizando a linguagem C. O Bonnie++ é uma versão reescrita do Bonnie, na linguagem C++, com algumas modificações que permitem a manipulação de arquivos de maior tamanho em sistemas de arquivos Reiser do Linux (COKER, 2000).

O intuito do *benchmark* Bonnie++ é efetuar uma sequência específica de operações de disco no ambiente testado, de maneira a medir o desempenho dessas operações. O programa parte do princípio de que o arquivo para testes tem um tamanho fixo pré-determinado pelo usuário. A primeira operação executada é a de criar o arquivo de testes através de gravação sequencial no disco. Depois, é efetuada uma leitura sequencial do arquivo criado, para finalmente efetuar uma série de leituras aleatórias sobre o mesmo. Todos os números de desempenho dessas operações são devolvidos pelo *benchmark* ao final da execução (BRAY, 1996).

Tanto o *benchmark* Bonnie original quanto o Bonnie++ são gratuitos, com licenças específicas criadas pelos seus autores. O *benchmark* Bonnie encontra-se disponível para *download* em *http://www.textuality.com/bonnie*, enquanto que o programa Bonnie++ está disponível para *download* em *http://www.coker.com.au/bonnie++*.

3.1.7 HINT – Hierarchical INTegration

A ferramenta HINT (*Hierarchical INTegration*) foi desenvolvida em 1995 por John Gustafson e por Quinn O. Snell, pesquisadores do SCL (*Scalable Computing Laboratory*) do *Ames Laboratory*, uma entidade ligada ao Departamento de Energia dos Estados Unidos. O princípio do HINT é o de que normalmente os computadores são velozes logo após a inicialização e vão ficando mais lentos durante a sua utilização. Tal queda de desempenho ocorre devido, principalmente, à diminuição da memória cache disponível, tornando-se necessária a utilização da memória RAM e, posteriormente, do disco rígido. Essas mudanças de comportamento são detectáveis com os resultados gerados pelo HINT.

De forma simplificada, o HINT efetua cálculos de integração numérica, de magnitude crescente, de maneira a induzir o sistema a esgotar a quantidade de memória disponível, obrigando a utilização da memória de nível imediatamente inferior, até um determinado limite de iterações. Ao contrário dos *benchmarks* tradicionais, o HINT não trabalha com problemas de tamanho fixo nem com os tempos para o cálculo do *workload*. Ao invés disso,

o software utiliza uma métrica chamada QUIPS (*Quality Improvement per Second*), que é determinado através da precisão com que a integração numérica é realizada a cada ciclo de teste. Dessa maneira, o HINT fornece resultados relativos ao desempenho de uma determinada máquina e tamanho de problema, demonstrando também o comportamento do equipamento conforme a memória disponível vai diminuindo.

O HINT é um *benchmark* escalável, facilmente portável para diversas arquiteturas. É possível executá-lo em praticamente qualquer equipamento que seja capaz de realizar aritmética digital, desde uma calculadora de bolso até um supercomputador (GUSTAFSON, SNELL, 1995). Atualmente, o projeto HINT é suportado pela *Brigham Young University*, que mantém um *site* com as informações gerais sobre o software e disponibiliza os códigos-fonte do mesmo, através do endereço *http://hint.byu.edu*.

3.1.8 Netperf

O *benchmark* Netperf foi desenvolvido em 1993 por Rick Jones, engenheiro da *Hewlett-Packard*. Desde sua primeira versão, o foco dos testes é a análise de redes entre sistemas Unix. Até hoje o Netperf é uma ferramenta muito popular entre os usuários desse sistema. O Netperf pode ser utilizado para medir a performance de várias tecnologias de redes, testando a taxa de transferência (*throughput*) unidirecional e a latência[12] entre os nós.

Atualmente, o Netperf é capaz de efetuar medições nos seguintes ambientes: TCP e UDP através de *sockets* BSD, tanto para IPv4 como para IPv6, DLPI[13], s*ockets* de domínios Unix e SCTP[14] para Ipv4 e Ipv6. O Netperf é gratuito e pode ser obtido através do endereço *http://www.netperf.org* (NETPERF, 2009).

3.1.9 VMmark

O software VMmark, produzido pela VMware, é uma das poucas ferramentas de *benchmark* atualmente disponíveis para testes específicos em máquinas virtuais. Seu funcionamento consiste basicamente no processamento de vários tipos de *workloads* que

[12] A latência pode ser medida de forma *unidirecional*, que é o tempo transcorrido entre o envio de um pacote de dados da fonte e seu recebimento pelo destino, ou *bidirecional*, que é o tempo transcorrido entre o envio do pacote pela fonte, o recebimento e devolução à fonte pelo destinatário e o recebimento do pacote pela fonte.

[13] *Data Link Provider Interface*, interface que corresponde à camada de dados do modelo OSI de sete camadas.

[14] *Stream Control Transmission Protocol*, protocolo da camada de transporte do modelo OSI de sete camadas.

visam simular diversas aplicações do equipamento testado, como o comportamento do sistema no papel de um servidor *web*, *firewall* ou de banco de dados, entre outros. Uma execução do VMmark dura aproximadamente três horas, nas quais o software coleta dados referentes ao desempenho do sistema a cada 60 segundos, sendo que ao final dessa execução todos esses dados são agrupados e processados conforme a característica de cada *workload* que foi executado, resultando em uma pontuação para cada *workload*. Por fim, todas as pontuações são normalizadas e o escore final é determinado através de uma média geométrica dos resultados normalizados.

Apesar do VMmark ser oferecido como um software gratuito, sua utilização requer uma estrutura de servidores de virtualização relativamente complexa, que deve possuir inclusive alguns softwares comerciais, como o SO *Windows Server 2003 R2 Enterprise Edition*, entre outros. O VMmark pode ser obtido para cópia através do site *http://www.vmware.com* (VMWARE, 2009a).

3.2 Resumo Comparativo das Ferramentas de *Benchmark*

Nesta seção, é apresentada uma breve síntese das características das ferramentas de *benchmark* descritas anteriormente. A Tabela 3 resume os principais aspectos de cada um dos pacotes descritos.

Tabela 3. Principais características dos pacotes de *benchmark* estudados

	Funcionalidades	**Objetivos**	**Licença**
SPEC CPU2006	Processamento de números inteiros e de ponto flutuante	Análise de desempenho do conjunto CPU, memória e compilador	Comercial (de US$ 200 até US$ 800)
HPL	Processamento de matrizes e cálculos de álgebra linear, utilizado no projeto TOP500	Análise de desempenho da CPU	Livre
Dhrystone	Execução de acessos à memória, cálculos com números inteiros e decisões lógicas simples	Análise de desempenho da CPU e da memória	Livre
STREAM	Execução de quatro operações comuns sobre vetores de grandes dimensões	Análise da largura de banda do subsistema de memória	Livre

Iometer	Medição de desempenho sobre *workloads* de I/O	Análise de desempenho de subsistemas de I/O	Livre
Bonnie++	Operações de leitura e gravação sequencial e randômica sobre arquivos	Análise de desempenho de I/O	Livre
HINT	Cálculos de integração numérica que simulam o comportamento do sistema conforme o uso de memória aumenta	Análise de desempenho da CPU, memória e I/O	Livre
Netperf	Utiliza dois nós de uma rede para medir a transmissão de pacotes	Análise de desempenho de subsistemas de rede	Livre
VMmark	Mede o desempenho sobre diversos *workloads*, cada um representando uma aplicação específica	Análise de desempenho da máquina virtual em aplicações distintas	Livre

Para o desenvolvimento deste trabalho, procurou-se dar preferência a softwares de *benchmark* de domínio público. Foram selecionados quatro tipos específicos de *benchmarks*: um pacote para avaliação do desempenho da CPU, outro para a avaliação da performance do disco e subsistemas de I/O, um software para medição do tráfego de rede e, finalmente, um *benchmark* para avaliação de desempenho da hierarquia de memória.

Para avaliação de desempenho do processador, optou-se pelo HPL, devido ao histórico de utilização deste software, inclusive na classificação TOP500. Para as avaliações de performance de I/O, foi selecionado o software Bonnie++, que foi criado com foco nessa atividade em ambiente Linux. Para os testes de desempenho de rede, utilizou-se o *benchmark* Netperf, amplamente utilizado em redes de sistemas Unix. Para análise da largura de banda de memória disponível para o sistema, escolheu-se o STREAM, que foi desenvolvido especificamente com este intuito. Por fim, selecionou-se também o *benchmark* HINT, para verificação do comportamento de cada ambiente de acordo com o aumento da utilização dos níveis de memória do equipamento.

4 Testes e Resultados Obtidos

Para a execução dos testes de desempenho sobre as ferramentas de virtualização abordadas neste trabalho, foi utilizado um notebook Dell Vostro 1000, equipado com processador *single-core* AMD Sempron 3600+ de 2.0 GHz (64 KB de memória *cache* L1 e 256 KB de *cache* L2), placa-mãe com chipset ATI Radeon Xpress 1150 e disco rígido de 80 GB com interface Serial-ATA de 150 MBps e velocidade de 4200 RPM.

Para os testes de desempenho em sistema Linux nativo (sem utilização de virtualização), utilizou-se somente 512 MB de memória RAM, padrão DDR2 de 667 MHz. Durante os testes das ferramentas de virtualização, foram instalados 2 GB de memória RAM padrão DDR2 de 667 MHz, para possibilitar a criação de máquinas virtuais com 512 MB de memória RAM sem correr-se o risco de tornar-se necessária a utilização de memória virtual.

Em todos os ambientes, o sistema operacional utilizado foi o Linux, na distribuição *SUSE Linux Enterprise Server* versão 10 SP1, com *kernel* versão 2.6.18.8 em arquitetura X86 (32 bits). Com o intuito de oferecer o ambiente nas mesmas condições para todos os testes, foram instalados os seguintes pacotes adicionais ao sistema Linux: *gcc* (compilador C), *g++* (compilador C++), *gfortran* (compilador Fortran) e *make* (gerador de executáveis). Esses pacotes são necessários para compilação dos *benchmarks* a serem executados e para a instalação dos aplicativos MPICH e ATLAS, que são pré-requisitos para a execução do *benchmark* HPL. Uma descrição mais detalhada de como foi efetuada a instalação e configuração de cada ambiente e software encontra-se no Anexo A.

Em todos os casos, cada *benchmark* foi executado cinco vezes, em sequência, tendo seus resultados gravados em arquivos de texto. Após foram realizados os cálculos dos resultados médios e do desvio padrão de cada ambiente testado. Os resultados atingidos pelo sistema executando o Linux nativo (sem virtualização) são tomados como referência para comparações entre as ferramentas. A seguir, estão apresentados os resultados médios obtidos em cada ambiente analisado.

4.1 *Benchmark* HPL

Uma única execução do *benchmark* HPL consiste em uma série de operações sobre matrizes. A cada matriz processada com sucesso, o algoritmo retorna, dentre outras informações, o tempo de processamento e o valor em *gigaflops* atingido pelo ambiente durante a execução. No Anexo A encontram-se maiores detalhes da configuração do HPL

para os testes. Na Tabela 4 são apresentados os resultados obtidos com o *benchmark* HPL nos ambientes testados.

Tabela 4. Resultados dos testes com o *benchmark* HPL

Ferramenta	GFlops (Média)	Desvio Padrão (%)	Perda (%)
Nenhuma (Linux nativo)	3,309	0,88	-
Citrix XenServer	3,225	0,18	2,54
VMware Server	2,965	0,16	10,42
Sun VirtualBox	2,949	2,45	10,88

Como pode ser observado na Tabela 4, o VirtualBox e o VMware Server apresentaram uma perda de desempenho próximo a 10%, devido ao fato de que, por implementar a técnica de máquina virtual de sistema, ambos dependem de um sistema operacional hospedeiro para funcionar, o que se traduz em maior custo computacional. Já o XenServer apresentou resultados próximos aos do sistema nativo (perda inferior a 3%), graças à utilização da técnica de paravirtualização. A seguir, a Figura 8 ilustra um comparativo gráfico dos resultados obtidos durante a execução do *benchmark* HPL nos ambientes testados.

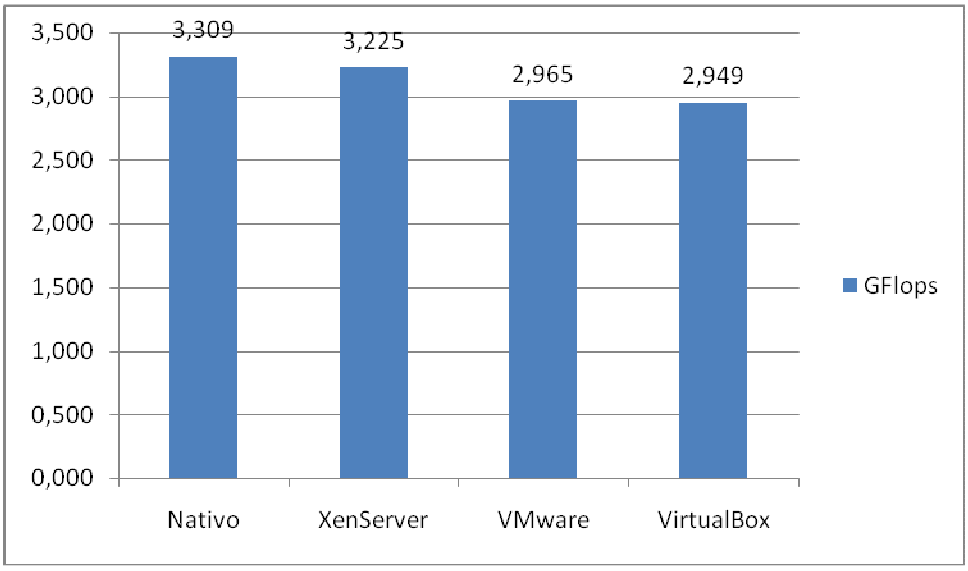

Figura 8. Média dos resultados do *benchmark* HPL em cada ambiente

4.2 *Benchmark* Bonnie++

O *benchmark* Bonnie++ efetua medições através da criação de um arquivo temporário em disco, que sofrerá uma série de operações de leitura e gravação sequenciais e aleatórias. Ao final, o software retorna medições com o desempenho obtido durante essas operações de I/O. Em cada ambiente, foram executados cinco testes em sequência, utilizando-se os parâmetros descritos no Anexo A.

Os números de cada teste estão detalhados separadamente para as operações de leitura sequencial, gravação sequencial e busca aleatória de dados no disco. A Tabela 5 mostra os resultados atingidos com o *benchmark* Bonnie++ durante os testes de leitura sequencial de dados. Observa-se novamente que o XenServer apresentou números muito próximos aos atingidos pela máquina sem virtualização, inclusive com melhora de desempenho de quase 10% no caso de leitura sequencial por blocos, enquanto que as demais ferramentas tiveram resultados similares. No caso da realização de operações de leituras sequenciais por caracteres, a perda de desempenho observada nessas ferramentas foi superior a 40%. Isso se deve, como já mencionado, ao fato de que tanto o VMware Server quanto o VirtualBox necessitam efetuar um tratamento de todas as operações solicitadas pelas VMs que estão em execução, enquanto que a arquitetura de paravirtualização implementada no XenServer exige o tratamento, por parte do hipervisor, apenas das instruções “sensíveis”, ou seja, instruções que possam causar exceções.

Tabela 5. Testes de leitura sequencial com o *benchmark* Bonnie++

	Por caracteres			**Por blocos**		
Ferramenta	**Kb/seg. (média)**	**Desvio Padrão (%)**	**Perda (%)**	**Kb/seg. (média)**	**Desvio Padrão (%)**	**Perda (%)**
Nenhuma (Linux nativo)	28.310,4	1,92	-	26.247,6	0,26	-
Citrix XenServer	27416,8	1,87	3,16	28775,8	4,72	-9,63
VMware Server	15993,0	4,01	43,51	20351,6	3,28	22,46
Sun VirtualBox	14.428,8	3,04	49,03	21714,4	1,07	17,27

A seguir, a Tabela 6 mostra os resultados atingidos em cada ambiente para as operações de gravação sequencial de dados no disco. Observa-se que o melhor desempenho foi atingido pelo XenServer, que nestes testes, do mesmo modo que no caso de operações de leitura, apresentou números superiores aos atingidos pela máquina nativa. Acredita-se que tal comportamento deve-se à estrutura reduzida do sistema XenLinux executado com o hipervisor no servidor, que oferece desempenho superior ao do Linux nativo. Por outro lado, as demais ferramentas tiveram desempenho parecido, com uma pequena vantagem para o VirtualBox na gravação sequencial por caracteres. No teste de gravação sequencial por blocos, tanto o VirtualBox quanto o VMware Server apresentaram perdas superiores a 50%.

Tabela 6. Testes de gravação sequencial com o *benchmark* Bonnie++

	Por caracteres			**Por blocos**		
Ferramenta	**Kb/seg. (média)**	**Desvio Padrão (%)**	**Perda (%)**	**Kb/seg. (média)**	**Desvio Padrão (%)**	**Perda (%)**
Nenhuma (Linux nativo)	26.208,8	5,09	-	27256,0	0,99	-
Citrix XenServer	29.569,2	7,93	-12,82	31.065,2	5,69	-13,98
VMware Server	11.222,0	4,02	57,18	13.203,0	5,11	51,56
Sun VirtualBox	12.803,0	2,14	51,15	13.050,0	3,50	52,12

Por fim, a Tabela 7 exibe os resultados atingidos em cada ambiente durante o teste de leitura aleatória em disco. Outra vez, a ferramenta XenServer atingiu resultados superiores aos apresentados pelo sistema nativo. O VMware Server apresentou significativa melhora no desempenho neste teste, com uma perda menor do que 4% em relação ao sistema nativo, enquanto que o VirtualBox também demonstrou melhora de performance, porém em menor grau, com uma diminuição de desempenho próxima a 22%. Tais comportamentos já eram esperados, devido novamente à estrutura de funcionamento de cada ferramenta e à vantagem obtida pela paravirtualização no que se refere ao tratamento de instruções “sensíveis”.

Tabela 7. Testes de leitura aleatória em disco com o *benchmark* Bonnie++

Ferramenta	Leituras / seg.	Desvio Padrão (%)	Perda (%)
Nenhuma (Linux nativo)	184,86	1,13	-
Citrix XenServer	189,22	6,05	-2,36
VMware Server	178,30	12,15	3,55
Sun VirtualBox	143,30	1,53	22,48

A Figura 9 ilustra graficamente os resultados apresentados nas Tabelas 5, 6 e 7. Nestes testes, onde o XenServer obteve performance superior à do sistema nativo, pode-se observar um comportamento que já fora detectado em situações semelhantes, em que ferramentas de virtualização apresentaram números melhores do que os sistemas nativos referenciados. Recentemente, a VMware anunciou que seu produto VMware Infrastructure 3 obteve resultados "recordes" em aplicações de virtualização de servidores web (VMWARE, 2009b), enquanto que a Citrix publicou resultados de testes sobre ambientes com o software Microsoft SQL Server 2008, onde seu produto XenServer 5 apresentou desempenho superior ao servidor nativo, com até 30% de vantagem para o XenServer sobre tal sistema (CITRIX, 2009b).

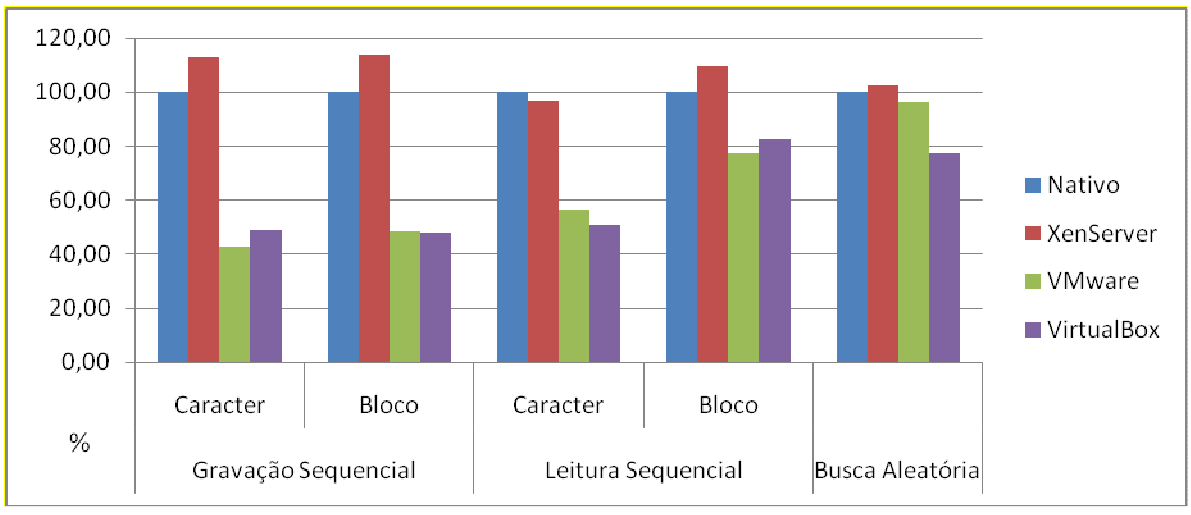

Figura 9. Resultados atingidos em cada ambiente com o *benchmark* Bonnie++ em relação ao ambiente sem virtualização

4.3 *Benchmark* Netperf

O *benchmark* Netperf efetua a medição da taxa de transferência de dados entre dois computadores através da rede. Um dos computadores executa um processo servidor chamado "*netserver"*, que é responsável por aguardar conexão do outro computador, que irá executar o processo cliente do *benchmark*, chamado *"netperf"*. No momento de cada chamada do cliente ao servidor, é possível determinar o tempo de duração do teste. Para as medições de todos os ambientes, foram efetuados cinco chamadas de 12 minutos em sequência, de maneira a completar uma hora contínua de teste.

Em todos os testes, o equipamento que atuou como servidor foi uma mesma máquina física independente, de maneira a oferecer igualdade de condições nos testes de todas as ferramentas de virtualização. Essa máquina consiste em uma estação equipada com processador *dual-core* AMD Athlon X2 de 2.1 GHz e 4 GB de memória RAM, conectada via rede *ethernet* de 100 Mbps e executando sistema operacional SUSE Linux Enterprise Server 10 SP1, com *kernel* versão 2.6.18.8 (versão para arquitetura x64). Tal estação esteve responsável por executar o processo "*netserver*", conforme descrito no Anexo A.

A Tabela 8 mostra os resultados atingidos com o *benchmark* Netperf nos ambientes testados. É possível observar que todas as ferramentas apresentaram números muito parecidos com os atingidos pelo sistema Linux nativo, com taxas de perda inferiores a 2%. Assim, observou-se que a execução desses testes sobre ambientes virtualizados teve pouca influência no desempenho na transferência de dados via rede.

Tabela 8. Resultados dos testes com o *benchmark* Netperf

Ferramenta de Virtualização	**Taxa de Transferência Média (x 10^6 bits/s)**	**Desvio Padrão (%)**	**Perda (%)**
Nenhuma (Linux nativo)	93,97	0,02	-
Sun VirtualBox	93,71	0,24	0,28
Citrix XenServer	93,09	0,03	0,94
VMware Server	92,58	0,01	1,48

A Figura 10 ilustra graficamente os resultados dos testes com o Netperf.

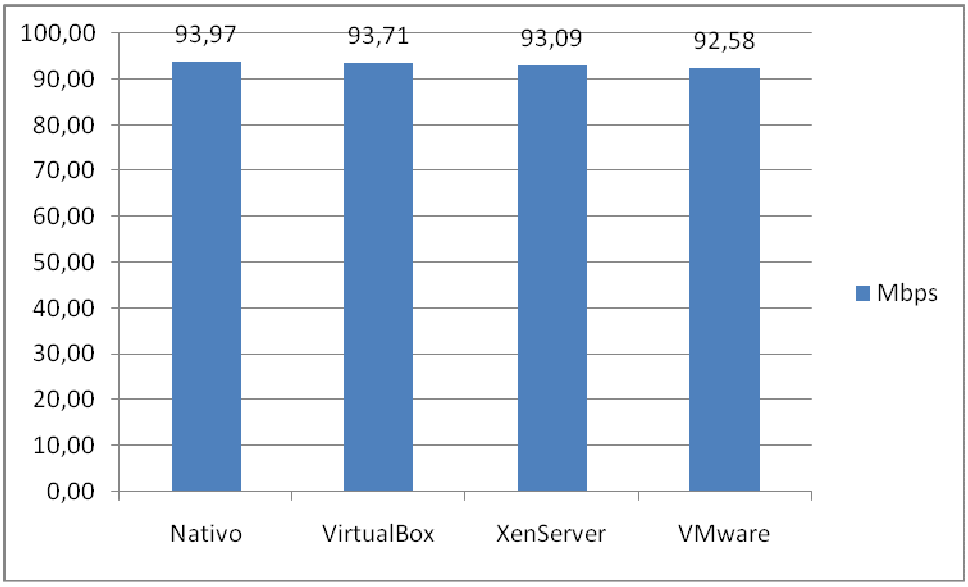

Figura 10. Médias dos resultados obtidos com o *benchmark* Netperf nos ambientes avaliados

4.4 *Benchmark* STREAM

O intuito do *benchmark* STREAM é determinar a largura de banda sustentável entre a memória e a FPU (*floating-point unit*, ou unidade de ponto flutuante) do equipamento testado, através de uma série de operações sobre vetores em memória. Para os testes com este *benchmark*, também foram efetuados cinco testes, em sequência, do STREAM em cada ambiente, conforme descrito no Anexo A.

Cada operação com vetores gera um conjunto de medições relativas à taxa de transferência de dados entre a memória e a FPU, cujos números são tratados separadamente dos demais. Para todas as operações, são efetuadas várias iterações da mesma operação, porém somente o melhor resultado atingido é considerado como índice de desempenho do *benchmark*. A primeira operação efetuada nesses testes é a operação *copy*, onde o STREAM efetua uma série de operações de cópia de matrizes em memória.

A Tabela 9 mostra os resultados atingidos nos ambientes testados durante a execução da operação *copy*. Todas as ferramentas testadas apresentaram alguma diminuição de performance, sendo que o VirtualBox obteve a maior perda de desempenho em relação ao sistema nativo, com uma redução superior a 12%. Já o XenServer o VMware Server demonstraram comportamento semelhante, com índices de perda de performance de aproximadamente 7% e 9%, respectivamente.

Tabela 9. Resultados dos testes com o *benchmark* STREAM na operação *copy*

Ferramenta de Virtualização	Taxa MB/s (média)	Desvio Padrão (%)	Perda (%)
Nenhum (Linux nativo)	2849,4727	0,30	-
Sun VirtualBox	2495,9071	4,12	12,41
Citrix XenServer	2647,7508	0,21	7,08
VMware Server	2599,3127	8,44	8,78

A seguir, a Tabela 10 mostra os resultados deste *benchmark* na operação *scale*. Essa operação é composta por uma série de operações de escalonamento de matrizes em memória. Nestes testes, o XenServer exibiu a menor redução de desempenho, próxima a 7%, contra perdas acima de 12% das demais ferramentas. Isto deve-se ao fato de que, neste caso, o XenLinux utilizado pelo XenServer ocupa uma parcela mínima dos recursos de memória do equipamento, enquanto que tanto o VirtualBox quanto o VMware Server exibem as consequências da dependência de um sistema hospedeiro completo, que naturalmente irá consumir mais recursos de hardware.

Tabela 10. Resultados dos testes com o *benchmark* STREAM na operação *scale*

Ferramenta de Virtualização	Taxa MB/s (média)	Desvio Padrão (%)	Perda (%)
Nenhum (Linux nativo)	2813,4518	0,27	-
Sun VirtualBox	2459,1261	4,93	12,59
Citrix XenServer	2589,2710	0,27	7,97
VMware Server	2420,5904	4,19	13,96

A seguir, a Tabela 11 mostra os resultados deste *benchmark* na operação *add*, que consiste em uma série de operações de adição de matrizes. Mais uma vez, a paravirtualização do XenServer demonstra maior eficiência na utilização do hardware. Os resultados obtidos através dessa ferramenta foram os que apresentaram menor perda de desempenho, inferiores a 10% em relação ao sistema nativo. As demais ferramentas retornaram números com perdas superiores a 15%.

Tabela 11. Resultados dos testes com o *benchmark* STREAM na operação *add*

Ferramenta de Virtualização	Taxa MB/s (média)	Desvio Padrão (%)	Perda (%)
Nenhum (Linux nativo)	3070,3940	0,37	-
Sun VirtualBox	2597,9628	6,22	15,39
Citrix XenServer	2774,2966	0,34	9,64
VMware Server	2528,9719	5,31	17,63

Finalmente, a Tabela 12 mostra os resultados deste *benchmark* na operação *triad*, composta por todas as operações efetuadas pelo *benchmark* - cópia, adição e escalonamento de matrizes. Neste caso, o XenServer obteve a menor redução de desempenho, com perda inferior a 9% em relação ao sistema nativo e isso se deve ao menor consumo de memória apresentado pelo XenLinux. As ferramentas VMware Server e VirtualBox tiveram desempenho semelhante, com perdas próximas a 15%.

Tabela 12. Resultados dos testes com o *benchmark* STREAM na operação *triad*

Ferramenta de Virtualização	Taxa MB/s (média)	Desvio Padrão (%)	Perda (%)
Nenhum (Linux nativo)	3037,0590	0,42	-
Sun VirtualBox	2577,5179	6,43	15,13
Citrix XenServer	2777,0600	0,34	8,56
VMware Server	2550,1864	3,27	15,27

A Figura 11 ilustra graficamente os resultados atingidos durante a execução do *benchmark* STREAM nos ambientes testados.

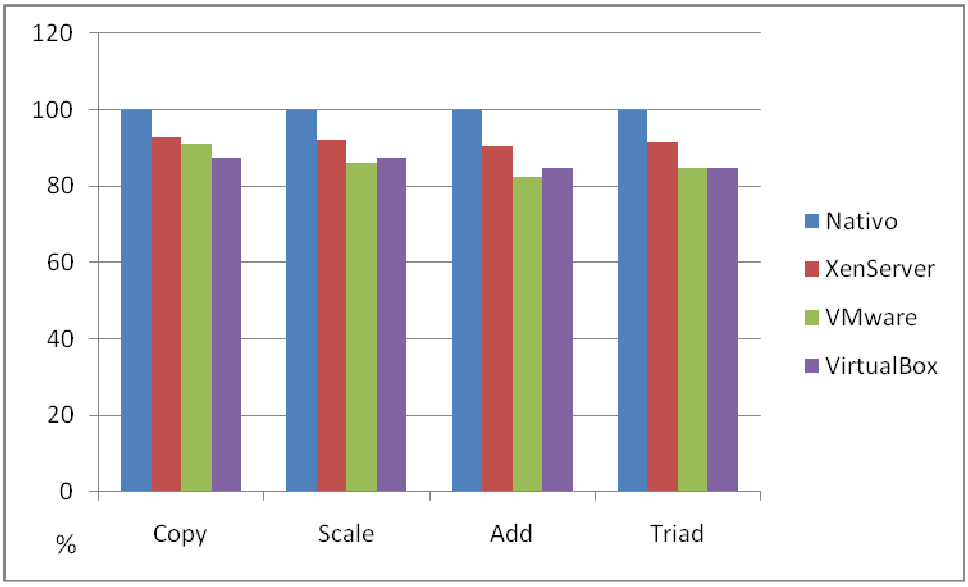

Figura 11. Resultados atingidos em cada ambiente com o *benchmark STREAM* em relação ao ambiente sem virtualização

4.5 *Benchmark* HINT

O objetivo principal do *benchmark* HINT é oferecer uma análise do comportamento do sistema testado quando os níveis de memória disponível diminuem, de forma a provocar uma diminuição no desempenho geral do sistema. Essa diminuição é semelhante àquela normalmente observada durante a utilização cotidiana do computador.

Da mesma maneira que os demais *benchmarks* executados, foram disparadas cinco execuções do HINT em sequência, para obter um conjunto de dados suficiente para o cálculo da média e do desvio padrão dos resultados. Os parâmetros utilizados para compilação e execução do *benchmark* HINT estão descritos no Anexo A.

Apesar de o HINT oferecer vários tipos de dados para os testes – por exemplo, com números de ponto flutuante, inteiros, duplos e outros – foram consideradas somente as avaliações com números inteiros, devido ao fato de que os testes com os outros tipos de dados disponíveis não apresentaram resultados confiáveis. No caso dos testes com números no formato *double* ou *float*, por exemplo, todos os ambientes apresentaram instabilidade, sendo que em algumas tentativas esses *benchmarks* foram executados com sucesso mas, em tentativas seguintes, esses testes não chegaram ao fim, deixando de responder ao sistema operacional. Como as execuções com números inteiros não apresentaram tais problemas, optou-se por utilizar somente esses resultados para fins de análise.

A Tabela 13 mostra os números obtidos com o *benchmark* HINT de números inteiros em cada ambiente. O desempenho do XenServer mostrou-se praticamente idêntico ao sistema nativo. O VMware Server e o VirtualBox, por sua vez, apresentaram resultados parecidos, com uma redução de cerca de 5% em relação ao sistema nativo. Novamente, credita-se essa diferença ao fato de haver um sistema operacional completo sendo executado sob o VMware Server e o VirtualBox, desta forma, tendo-se um maior consumo de recursos. Como o XenServer é executado sobre o XenLinux, que é extremamente enxuto no que se refere ao consumo de recursos de hardware, era esperado que os resultados desta ferramenta fossem superiores.

Tabela 13. Resultados dos testes com o *benchmark* HINT com números inteiros

Ferramenta de Virtualização	Índice QUIPS (média)	Desvio Padrão (%)	Perda (%)
Nenhum (Linux nativo)	145071545,869025	0,10	-
Citrix XenServer	145177030,066370	0,58	-0,07
Sun VirtualBox	138653564,862961	0,53	4,42
VMware Server	138062503,729182	0,52	4,83

A Figura 12 exibe graficamente os resultados obtidos durante os testes com o *benchmark* HINT nos ambientes analisados.

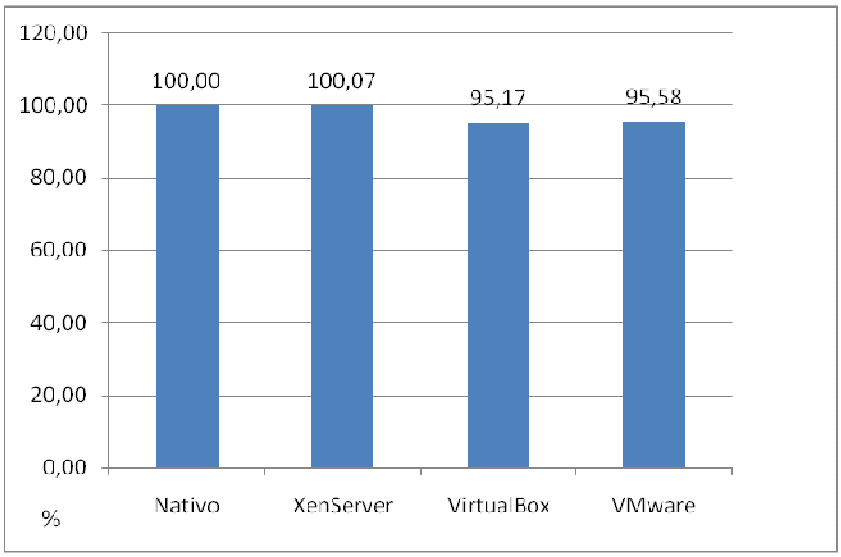

Figura 12. Média dos resultados de cada ambiente com o *benchmark* HINT com números inteiros

4.6 Considerações Finais

Com a execução dos *benchmarks* citados nesta seção, obteve-se números que permitiram cálculos de perda ou ganho de performance de cada ferramenta em relação a um ambiente não virtualizado. Em determinados aspectos, todas as ferramentas apresentaram desempenhos semelhantes, especialmente no que se refere à comunicação de rede, onde o pior resultado encontrado teve perda menor do que 2% em relação ao ambiente nativo.

Porém, é necessário destacar os números de desempenho obtidos pela ferramenta XenServer durante os testes. Em nenhum dos *benchmarks* efetuados esse software obteve perdas maiores do que 10%, sendo que nos testes de I/O a ferramenta apresentou números melhores do que aqueles obtidos pelo ambiente nativo. Isso evidencia a eficiência da técnica de paravirtualização quanto à utilização do hardware físico.

Por outro lado, a solução da Citrix requer uma estrutura física mais elaborada para seu funcionamento, inclusive necessitando de uma estação que executa o sistema operacional Microsoft Windows para instalação da ferramenta de gerenciamento XenCenter. Além disso, a própria configuração do software para funcionamento no ambiente físico é mais complexa do que as demais ferramentas testadas. Uma vez ajustada ao ambiente, o processo de criação e manutenção de máquinas virtuais é relativamente simples. Portanto, acredita-se que o XenServer é uma solução apropriada para aplicações de maior porte, como a virtualização de considerável número servidores para consolidação de recursos. Nessas aplicações, é comum que a ferramenta seja configurada apenas uma vez, durante a implantação do ambiente, e que mais tarde sejam adicionados mais recursos de hardware.

Já o VMware Server e o VirtualBox parecem ser alternativas igualmente interessantes em aplicações de menor porte, ou seja, quando não existe a necessidade de implantação de um grande número de servidores para virtualização. Ambas as ferramentas apresentaram desempenho parecido, mas não dispõem de recursos interessantes para ambientes de maior porte, como a possibilidade de *live migration* das máquinas virtuais. Uma vantagem do VMware Server é o fato de seu painel de controle ser acessível através de um navegador de internet comum, bastando acessar no *browser* o endereço IP do computador que está executando as máquinas virtuais. Essa opção permite que as VMs sejam gerenciadas tanto através do próprio equipamento que as executa quanto através de qualquer outra estação da rede.

No caso do VirtualBox, o controle da ferramenta e das VMs criadas dentro dela é efetuado através de uma interface gráfica local. Existe apenas a opção de acesso à

máquina virtual desejada via VRDP (*VirtualBox Remote Desktop Protocol*), que é uma extensão do protocolo Microsoft RDP (*Remote Desktop Protocol*) para as VMs do VirtualBox. O acesso via VRDP permite acessar, a partir de outro computador, a tela da máquina virtual desejada, possibilitando a sua utilização sem a necessidade de interação com a interface gráfica do VirtualBox, mas essa solução não permite gerenciamento completo da VM. Por outro lado, o VirtualBox oferece um recurso que está ausente no VMware Server, que é o acesso às pastas compartilhadas (*shared folders*). Essa opção encontra-se disponível somente em outros produtos da VMware, como por exemplo o vSphere ou o ESX.

Considerando-se o desempenho obtido nos testes realizados, é possível deduzir que o XenServer, entre as três ferramentas avaliadas, é a opção mais vantajosa para aplicação em ambientes de virtualização. Além disso, os recursos de gerenciamento oferecidos pelo software XenCenter – que oferece controle das máquinas virtuais e servidores físicos através de uma máquina específica para essa tarefa – e a opção de *live migration* das máquinas virtuais, entre outros itens, são de grande valia durante a implementação em casos reais de uso.

A ferramenta VirtualBox, na versão testada, não oferece o recurso de *live migration* de máquinas virtuais, o que limita a sua utilização a ambientes de menor porte, apesar de dispor da opção de criação de *snapshots* das máquinas virtuais. Porém, já estão disponíveis versões *beta* do software (tais como a VirtualBox 3.1.0 Beta 1), onde encontra-se o recurso de migração “quente” de máquinas virtuais, intitulado pela fabricante de “*Teleportation*” (SUN, 2009b).

O VMware Server, na versão testada, também não oferece o recurso de *live migration* de máquinas virtuais, da mesma forma que o VirtualBox. Porém, no caso deste software, a fabricante optou por oferecer tal recurso (chamado *VMotion*) somente nos seus produtos comerciais, tais como o ESX e o vSphere, desenvolvidos para uso em ambientes de médio/grande porte e de maior número de equipamentos físicos.

No próximo capítulo, serão abordados aspectos do recurso de migração “quente” de máquinas virtuais, que está disponível na ferramenta XenServer. As demais ferramentas não foram testadas devido ao fato de não oferecerem essa modalidade de migração nas versões avaliadas.

5 Migração de Máquinas Virtuais com XenServer

Neste capítulo, serão abordados os requisitos necessários e os resultados obtidos nos testes de migração “quente” (*live* migration) de máquinas virtuais com a ferramenta Citrix XenServer. Os outros softwares testados durante este trabalho não puderam ser avaliados quanto a este recurso por não oferecerem-no nas suas versões gratuitas. No caso da versão testada do VirtualBox, não existe essa opção de migração. Já a VMware comercializa um recurso de migração “quente” de máquinas virtuais intitulado *VMotion*, porém ele só está disponível como um pacote adicional para os produtos comerciais da empresa, como o ESX e o *vSphere*, ou seja, essa opção não existe para o VMware Server.

Como requisitos mínimos, o XenServer necessita de pelo menos dois servidores devidamente configurados em um mesmo *pool* de recursos. Um *pool* nada mais é do que um conjunto de servidores configurados para compartilhar determinados recursos. Ao criar um *pool*, alguns recursos avançados do XenServer estarão disponíveis para os servidores que o compõem, entre eles o *XenMotion*, que é responsável pela migração “quente” de máquinas virtuais. Para que a migração seja possível, é necessário que os servidores de origem e destino tenham arquitetura idêntica (no caso, mesmo processador), caso contrário, o software impede a operação. Além disso, é obrigatório que a máquina virtual a ser migrada não esteja fazendo uso de recursos “locais”, como discos virtuais criados sobre armazenamento local no servidor; portanto, faz-se necessário que as VMs a serem migradas estejam instaladas em discos virtuais armazenados em volumes compartilhados, tais como diretórios NFS[15]. Por fim, para que o XenServer ofereça a opção de *live migration*, cada VM deve ser configurada com as extensões de software do XenServer, chamadas de *XenServer Tools*, que configuram alguns serviços adicionais no sistema hóspede.

Após proceder com a instalação do software XenServer em cada servidor, é necessário adicioná-los ao mesmo *pool*, o que pode ser feito rapidamente através da ferramenta XenCenter. Deve-se então configurar um volume compartilhado na ferramenta XenCenter, que tanto pode ser um volume compartilhado da rede através de NFS, quanto um equipamento de *storage* de rede conectado por barramentos *iSCSI* ou *Fibre Channel*, ou ainda *storages* conectados através de hardware, com adaptadores HBA[16] dos fabricantes Emulex, QLogic ou compatíveis. A partir desse momento, basta criar uma nova máquina

[15] *Network File System*, ou sistema de arquivos de rede, um protocolo de compartilhamento de diretórios utilizado por sistemas operacionais como o Linux.

[16] *Host Bus Adapters*, componentes de hardware que servem de adaptadores para conexão de equipamentos de *storage*.

virtual, indicando como dispositivo de armazenamento o volume compartilhado previamente configurado. Uma descrição detalhada dos procedimentos que foram adotados para configuração do XenServer no ambiente avaliado encontra-se no Anexo B.

Uma vez configurado o ambiente e criada a máquina virtual, o processo de migração da VM entre servidores é simples, podendo ser efetuado tanto através da ferramenta XenCenter quanto através de comandos no XenLinux de qualquer estação com o XenServer. Para efetuar a migração com o XenCenter, basta selecionar a máquina virtual desejada, acessar o menu “VM / Migrate to Server” e escolher o servidor de destino desejado para a migração.

O processo de *live migration* é então iniciado, mas os processos que estão sendo executados sobre a VM sofrem intereferência muito pequena. No momento da migração, a VM sofre uma pausa momentânea no seu processamento, para que todos os seus processos sejam movidos para outra estação física do XenServer, onde terão continuidade. No caso dos testes realizados, o processo completo do XenMotion leva aproximadamente um minuto, mas a pausa no processamento durante a migração é quase imperceptível ao usuário, não causando maiores transtornos na utilização da VM. O tempo gasto durante essa pausa, que nos testes realizados não chegou a um segundo, é referente à efetiva troca de servidor que executa a máquina virtual.

Durante este trabalho, o processo de migração com a ferramenta XenServer foi avaliado em um ambiente de dois servidores HP dx5150 MT executando o XenServer, mais uma estação com sistema Microsoft Windows XP Professional SP3 executando o XenCenter. Ambos os servidores possuíam configuração idêntica, sendo que cada um deles era composto por um processador AMD Athlon 64 3500+ de 2.2 GHz (64 KB de memória *cache* L1 e 512 KB de *cache* L2), placa-mãe com *chipset* ATI Radeon Xpress 200, 1 GB de memória RAM, padrão DDR de 400 MHz, e disco rígido de 80 GB com interface Serial-ATA de 150 MBps e velocidade de 5400 RPM. Os servidores e a estação do XenCenter estavam interligados através de rede *ethernet* de 100 Mbps. Para armazenamento, optou-se por configurar um novo volume de 66 GB em um dos servidores, compartilhado através de NFS.

Neste ambiente, foi criada uma máquina virtual para execução do sistema operacional SUSE Linux Enterprise Server 10 SP1, configurada com 1 processador virtual e 512 MB de memória RAM. Como disco rígido virtual da VM, foi configurado um novo disco de 8 GB a ser armazenado no volume compartilhado por NFS previamente configurado. Após a instalação do sistema operacional Linux na VM, foi instalado o pacote *XenServer Tools*, que possibilita a migração “quente” da VM entre os servidores.

Como o primeiro dos testes para avaliação de desempenho da máquina virtual, optou-se por efetuar compilações do pacote de software ATLAS, que oferece ferramentas para processamento de álgebra linear (ATLAS, 2009). Esse pacote pode ser obtido no endereço *http://math-atlas.sourceforge.net*. Nesse caso, foram efetuadas três compilações do ATLAS na máquina virtual, sem efetuar migração entre os servidores, para determinar o tempo médio necessário para compilação em condições normais. Em seguida, foram executadas outras três compilações do ATLAS, efetuando-se durante cada compilação uma única migração da VM entre os servidores, de maneira a verificar o impacto da migração da máquina virtual no tempo de compilação.

A Tabela 14 mostra os tempos médios de compilação obtidos em cada avaliação. Observa-se que a migração da máquina virtual entre os servidores exerceu pequena influência no tempo necessário para compilação do ATLAS, causando uma perda menor do que 5%.

Tabela 14. Tempos para compilação do pacote de software ATLAS

	Sem migração		**Efetuando migração**		
	Tempo médio (s)	**Desvio padrão (%)**	**Tempo médio (s)**	**Desvio padrão (%)**	**Perda (%)**
Real	946,69	2,50	990,62	1,67	4,64

A Figura 13 mostra graficamente a média dos tempos gastos durante os testes sem migração e efetuando-se a migração da VM.

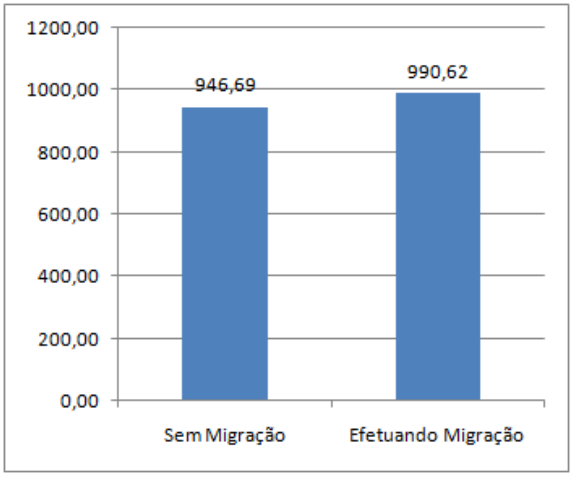

Figura 13. Médias dos tempos consumidos durante a compilação do pacote ATLAS

Em outra série de testes, executou-se um programa na linguagem C que efetua o cálculo de multiplicação de matrizes. Foram executados cálculos sobre matrizes de três ordens diferentes: 2.000, 2.500 e 3.000. Para cada matriz, foram feitos três cálculos sequenciais, sem a migração da VM entre servidores. A seguir, foram efetuados novamente os mesmos três cálculos, porém, durante cada cálculo foi efetuada uma migração da VM entre os servidores disponíveis.

A Tabela 15 mostra os tempos médios obtidos nos cálculos das matrizes citadas. Observa-se que, novamente, o processo de migração causou pouco atraso no processamento, mas esse custo computacional causa maiores prejuízos em processos menores, ou seja, quanto maior a matriz, menor foi o impacto da migração sobre o seu processamento. Para a matriz de ordem 2.000, a perda de performance foi de quase 28%, um número considerável. Já para a matriz de ordem 2.500, essa diminuição ficou em cerca de 12%, que já pode ser considerada uma taxa aceitável. Finalmente, para a matriz de ordem 3.000, a perda foi ainda menor, próxima a 8%.

Tabela 15. Tempos para cálculo da multiplicação de matrizes

Ordem	Sem migração		Efetuando migração		
	Tempo médio (s)	Desvio padrão (%)	Tempo médio (s)	Desvio padrão (%)	Perda (%)
2.000	195,54	0,31	249,87	0,07	27,78
2.500	409,94	0,71	459,41	0,35	12,07
3.000	703,29	0,62	759,73	0,08	8,03

Fica evidente que o tempo gasto na migração de máquinas virtuais torna-se praticamente imperceptível, conforme aumentam os tempos de utilização da CPU da máquina virtual. Neste contexto, pode-se concluir que o recurso XenMotion de migração “quente” de máquinas virtuais da ferramenta XenServer possibilita redundância e balanceamento da carga entre os servidores físicos e torna viável a aplicação da ferramenta em aplicações onde a disponibilidade dos serviços é crítica, uma vez que o tempo de processamento adicional causado pelo processo de migração é relativamente pequeno para processos mais pesados. Por outro lado, eventuais processos que apresentem um menor consumo de tempo de CPU e estejam rodando na máquina virtual durante a migração irão apresentar maiores perdas, devido ao fato de que o tempo mínimo necessário para

migração de uma máquina virtual é sempre o mesmo. No caso das máquinas utilizadas em todos os testes, esse tempo ficou próximo a um minuto, para o processo de migração de uma VM configurada com um processador virtual, 512 MB de memória RAM e disco rígido de 8 GB armazenado em um volume NFS compartilhado na rede.

A Figura 14 exibe graficamente os tempos médios obtidos no cálculo das matrizes sem o processo de migração e durante a migração das máquinas virtuais.

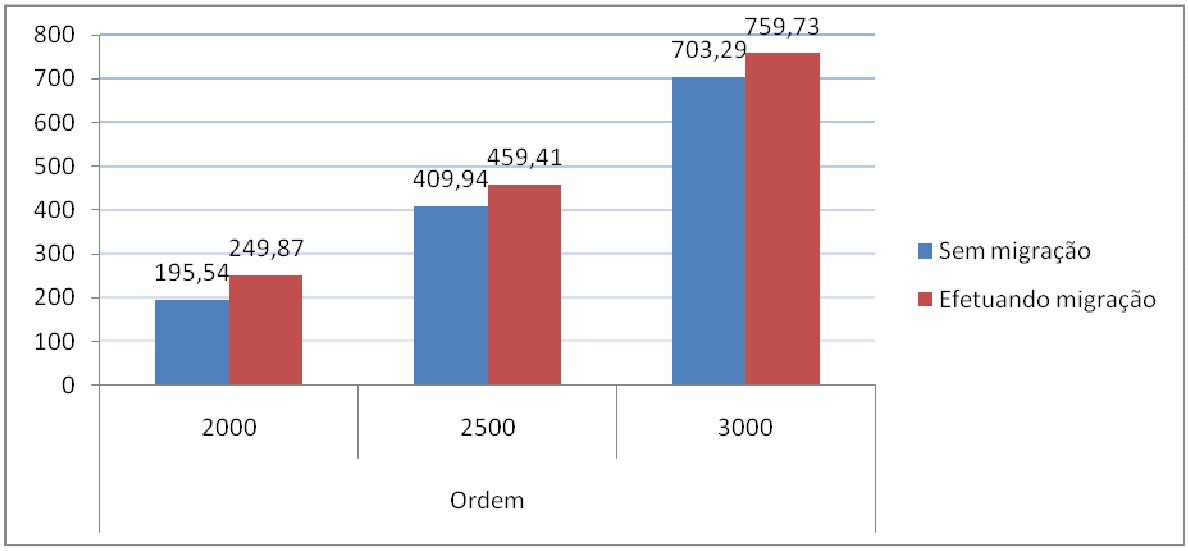

Figura 14. Médias dos tempos consumidos para cálculo de matrizes de várias ordens

5.1 Considerações Finais

Através dos testes efetuados com o recurso *XenMotion* do XenServer, foi possível deduzir que a possibilidade de *live migration* de máquinas virtuais é de extrema importância no que se refere à capacidade de gerenciamento dos recursos de hardware disponíveis. Os números obtidos indicam que o custo representado pelo tempo para migração das VMs é pequeno, se comparado ao benefício obtido pelo balanceamento da carga entre os servidores disponíveis, especialmente pelo fato de não haver a necessidade da parada no processamento das máquinas virtuais.

A presença dessa opção é mais um indício de que o XenServer é uma ferramenta projetada para implantação em ambientes de produção, mesmo em sua versão gratuita. As demais ferramentas não foram testadas por não oferecerem o recurso de migração “quente”.

6 Conclusão

Neste trabalho, foi efetuada uma avaliação sobre a viabilidade do uso de técnicas de virtualização em ambientes computacionais de alto desempenho. Para atingir tal objetivo, estudou-se os conceitos básicos de virtualização, bem como a classificação das máquinas virtuais segundo sua estratégia de implementação – emulação, máquina virtual de processo e máquina virtual de sistema (virtualização total ou paravirtualização).

Também foram analisadas e testadas três das principais ferramentas de virtualização disponíveis no mercado: Sun VirtualBox, Xen e VMware Server, detalhando-se as características técnicas e principais recursos de cada uma delas. Foram, ainda, estudados os principais métodos de *benchmark* utilizados no mercado, bem como os objetivos e modo de funcionamento de cada metodologia. Após, selecionou-se cinco pacotes de *benchmark* para a avaliação de desempenho das ferramentas de virtualização citadas. Adiante, foram efetuados testes de *live migration* de máquinas virtuais com a ferramenta Citrix XenServer, a única ferramenta avaliada que oferece tal recurso em sua versão gratuita. Durante esses testes, procurou-se determinar qual a magnitude da perda de desempenho causada pela migração de máquinas virtuais.

Neste contexto, é possível concluir que as técnicas de virtualização podem ser utilizadas em ambientes computacionais de alto desempenho, desde que tal tarefa seja cuidadosamente planejada e que a ferramenta de virtualização selecionada seja adequada à aplicação desejada. As possibilidades de criação de *snapshots*, bem como de migração “quente” de máquinas virtuais entre os servidores do ambiente, entre outros recursos, têm potencial para aumentar consideravelmente os índices de disponibilidade dos equipamentos, bem como facilitam o gerenciamento da carga dos computadores da rede, dentre outros benefícios. Por outro lado, ao mesmo tempo que algumas ferramentas ofereceram bons números de desempenho, outras apresentaram índices de perda de performance consideráveis, que não podem ser relevados durante o processo de seleção da ferramenta para implantação.

Durante o desenvolvimento deste trabalho, foram avaliadas três ferramentas disponíveis no mercado: Sun VirtualBox, VMware Server e Citrix XenServer. As duas primeiras apresentaram resultados próximos entre si, obtendo maiores índices de perda de desempenho em aplicações que exigem muitas operações de I/O, como demonstrado através do *benchmark* Bonnie++. Tal comportamento também foi verificado em aplicações onde era exigida maior utilização de CPU, como demonstrado pelo *benchmark* HPL. Já o XenServer apresentou números muito próximos aos do sistema nativo (sem utilização de

virtualização). Em alguns casos, inclusive, o desempenho dessa ferramenta foi superior ao do sistema nativo. Esse desempenho deve-se em grande parte à técnica de paravirtualização implementada neste software, que permite que seu sistema operacional XenLinux apresente menor consumo de recursos de hardware.

Os testes e experimentos efetuados com o XenServer no que se refere à *live migration* de máquinas virtuais demonstraram a aplicabilidade dessa opção em ambientes de alto desempenho, uma vez que o custo computacional envolvido nessa operação é pequeno, sendo perceptível em situações onde a máquina virtual executa processos de pequena carga computacional. Em situações reais, onde vários computadores formam agregados de alto desempenho, o custo do tempo gasto durante a migração é superado pelo benefício do aumento da disponibilidade das máquinas e da ampliação das possibilidades de gerenciamento da carga computacional entre os equipamentos. Como desvantagem, pode-se citar a necessidade de um volume compartilhado ou equipamento de *storage* que disponibilize espaço suficiente para a criação das máquinas virtuais. No caso de ter-se um grande número de VMs, o espaço exigido para seu armazenamento é elevado.

Através desses dados, pode-se concluir que tanto o VirtualBox quanto o VMware Server são ferramentas interessantes para atividades onde o desempenho da máquina virtual não é um fator crítico, pois a perda de performance em qualquer um desses softwares é considerável. Por outro lado, a facilidade para configuração e utilização dessas ferramentas é um ponto positivo, mostrando que esses softwares são apropriados para ambientes de teste e desenvolvimento. O desempenho e os recursos de gerenciamento apresentados pelo XenServer indicam que essa ferramenta é apropriada para ambientes onde a performance é um fator crucial para o sucesso da implementação. A contrapartida desse software são os requisitos necessários para a criação de um ambiente virtualizado, bem como a complexidade para sua configuração inicial. Essas características mostram que o XenServer deve ser considerado como uma opção para cenários de produção, onde o ambiente é configurado de maneira cuidadosamente planejada, possibilitando a expansão dos recursos com certa facilidade. Tais conclusões também são válidas para ambientes computacionais de alto desempenho, onde além da performance, a disponibilidade e a capacidade de gerenciamento dos recursos físicos são fatores de extrema importância.

6.1 Trabalhos Futuros

Por falta de tempo hábil no desenvolvimento deste trabalho, não foi possível configurar um ambiente de alto desempenho propriamente dito, com várias máquinas

interligadas entre si formando um agregado. Como trabalhos futuros, sugere-se a implantação do XenServer em um ambiente com número razoável de computadores, de maneira a criar um *pool* com boa quantidade de recursos de hardware disponíveis, para então criar uma série de máquinas virtuais nesse cenário, possibilitando a interligação dessas máquinas virtuais para formar um agregado. Tal ambiente ofereceria condições muito interessantes para testes de desempenho em situações reais de utilização de alto desempenho.

Também é possível sugerir que sejam efetuados os testes deste trabalho em outras ferramentas de virtualização disponíveis no mercado, especialmente as ferramentas comerciais da VMware, que prometem desempenho parecido ou superior ao nativo (VMWARE, 2009b) e que poderiam gerar resultados para comparação com a performance do XenServer. Ainda, seria interessante efetuar testes de desempenho na próxima versão do VirtualBox, que provavelmente irá oferecer o recurso de migração de VMs (chamado *Teleporting*), bem como na ferramenta de virtualização Hyper-V, incluída no sistema operacional Windows Server 2008, da Microsoft.

7 Referências Bibliográficas

AUTOMATICALLY TUNED LINEAR ALGEBRA SOFTWARE (ATLAS). **ATLAS.** 2009. Disponível em: <http://math-atlas.sourceforge.net>. Acesso em: 12 de set. 2009.

BRAY, Tim. **Bonnie.** 2009. Disponível em: <http://www.textuality.com/bonnie>. Acesso em: 12 de set. 2009.

CARISSIMI, Alexandre. Virtualização: da teoria a soluções. **26º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos,** Porto Alegre, RS, P. 173 – 207, 2008.

CARISSIMI, Alexandre. Virtualização: princípios básicos e aplicações. **9ª Escola Regional de Alto Desempenho (ERAD 2009),** Caxias do Sul, RS, P. 39 – 69, 2009.

CITRIX SYSTEMS, INC. **Citrix XenServer visão geral do produto.** Las Vegas, NV, Estados Unidos, 2008. Disponível em: <http://www.gandalph.com.br/img/Downloads/Citrix-XenServer-Port.pdf>. Acesso em: 27 de ago. 2009.

CITRIX SYSTEMS, INC. (2009a) **Xen.org.** Las Vegas, NV, Estados Unidos, 2009. Disponível em: <http://www.xen.org>. Acesso em: 20 de ago. 2009.

CITRIX SYSTEMS, INC. (2009b) **Virtualizing Microsoft SQL Server 2008 with Citrix XenServer.** Fort Lauderdale, FL, Estados Unidos, 2009. Disponível em: <http://www.citrix.com/-site-resources-dynamic-salesdocs-SQL_Server2008_XS5_WP.pdf>. Acesso em: 8 de nov. 2009.

COKER, Russell. **Bonnie++ now at 1.03e (last version before 2.0)!.** 2009. Disponível em: <http://www.coker.com.au/bonnie++>. Acesso em: 12 de set. 2009.

FEDERAL ELECTRONICS CHALLENGE (FEC). **Answers to frequent questions: total cost of ownership.** Washington, DC, Estados Unidos, 2007. Disponível em: <http://www.federalelectronicschallenge.net/resources/docs/costofown.pdf>. Acesso em: 29 de set. 2009.

FREE SOFTWARE FOUNDATION. **GNU General Public License Version 2, June 1991.** Boston, MA, Estados Unidos, 1991. Disponível em: <http://www.gnu.org/licenses/gpl-2.0.html>. Acesso em: 20 de ago. 2009.

GRAY, Jim. **Database and transaction processing performance handbook.** Maynard, MA, Estados Unidos, 1993. Disponível em: <http://research.microsoft.com/en-us/um/people/gray/benchmarkhandbook/chapter1.pdf>. Acesso em: 7 de set. 2009.

GOTH, Greg. Virtualization: old technology offers huge new potential, **IEEE Distributed Systems Online,** V. 8, N. 2, 2007, art. n. 0702-o2003.

GUSTAFSON, John L.; SNELL, Quinn O. HINT: A new way to measure computer performance. **Proceedings of the 28th Annual Hawaii International Conference on System Sciences (HICSS'95)**, Washington, DC, Estados Unidos, P. 392 – 401, 1995.

IBM CORP. **IBM systems virtualization.** Armonk, NY, Estados Unidos, 2005. Disponível em: <http://publib.boulder.ibm.com/infocenter/eserver/vlr2/topic/eicay/eicay.pdf>. Acesso em: 11 de ago. 2009.

IOMETER. **Iometer.** 2009. Disponível em: <http://www.iometer.org>. Acesso em: 12 de set. 2009.

JONES, M. Tim. **Virtual Linux.** Costa Mesa, CA, Estados Unidos, 2006. Disponível em: <http://www.ibm.com/developerworks/library/l-linuxvirt>. Acesso em: 11 de jul. 2009.

KAKULAVARAPU, Prasad. **Linux kernel compiling.** Disponível em: <http://software.intel.com/en-us/articles/linux-kernel-compiling>. Acesso em: 31 de ago. 2009.

KS, Hm Ar. **Design methodology for development of behavioral synthesis generic and synthetic benchmarks.** Los Angeles, CA, Estados Unidos, 1996. Disponível em: <ftp://ftp.cs.ucla.edu/tech-report/1996-reports/960>. Acesso em: 7 de set. 2009.

LAUREANO, Marcos Aurelio Pchek.; MAZIERO, Carlos Alberto. Virtualização: conceitos e aplicações em segurança. **Simpósio Brasileiro de Segurança da Informação e Sistemas Computacionais 2008 (SBSEG 2008)**, Gramado, RS, 2008.

MADDEN, Brian. **Emulation, paravirtualization and pass-through: what you need to know for client hypervisors.** Disponível em: <http://www.brianmadden.com/blogs/brianmadden/archive/2009/08/10/emulation-paravirtualization-and-pass-through-what-you-need-to-know-for-client-hypervisors.aspx>. Acesso em: 11 de ago. 2009.

MCCALPIN, John D. **STREAM: sustainable memory bandwidth in high performance computers.** Estados Unidos, 2009. Disponível em: <http://www.cs.virginia.edu/stream>. Acesso em: 19 de set. 2009.

NETLIB. **LAPACK – Linear Algebra PACKage.** Estados Unidos, 2009. Disponível em: <http://www.netlib.org/lapack>. Acesso em: 12 de set. 2009.

NETPERF. **The Netperf Homepage.** Estados Unidos, 2009. Disponível em: <http://www.netperf.org>. Acesso em: 19 de set. 2009.

NOVELL, INC. **Harness the power of virtualization for server consolidation.** Estados Unidos, 2009. Disponível em: <http://www.novell.com/rc/docrepository/public/37/RC-23126/4622022_en.pdf>. Acesso em: 18 de set. 2009.

POPEK, Gerald J.; GOLDBERG, Robert P. Formal requirements for virtualizable third generation architectures, **Communications of the ACM,** Los Angeles, CA, Estados Unidos, V. 17, N. 7, P. 412 – 421, jul. 1974.

PRATT, Ian et. al. **Xen and the art of virtualization.** Cambridge, Inglaterra, 2004. Disponível em: <http://www.cl.cam.ac.uk/research/srg/netos/papers/2004-xen-ols.pdf>. Acesso em: 27 de ago. 2009.

ROSE, Robert. **Survey of system virtualization techniques.** Bend, OR, Estados Unidos, 2004. Disponível em: <http://www.robertwrose.com/vita/rose-virtualization.pdf>. Acesso em: 11 de jul. 2009.

SILBERSCHATZ, Abraham; GAGNE, Greg; GALVIN, Peter Baer. **Fundamentos de sistemas operacionais.** 6. ed. Rio de Janeiro: Campus, 2004. 600 p.

STANDARD PERFORMANCE EVALUATION CORPORATION (SPEC). **Standard Performance Evaluation Corporation.** Warrenton, VA, Estados Unidos, 2009. Disponível em: <http://www.spec.org>. Acesso em: 31 de ago. 2009.

SUN MICROSYSTEMS, INC. (2009a) **VirtualBox.** Santa Clara, CA, Estados Unidos, 2009. Disponível em: <http://www.virtualbox.org>. Acesso em: 20 de ago. 2009.

SUN MICROSYSTEMS, INC. (2009b) **VirtualBox 3.1.0 Beta 1 released.** Disponível em: <http://forums.virtualbox.org/viewtopic.php?f=15&t=24458>. Acesso em: 25 de nov. 2009.

TANENBAUM, Andrew S. **Sistemas operacionais modernos.** 2. ed. São Paulo: Prentice Hall, 2007. 712 p.

TOP500.ORG. **The Linpack Benchmark.** Disponível em: <http://www.top500.org/project/linpack>. Acesso em: 15 de set. 2009.

TORRES, Gabriel. Athlon XP vs. Pentium 4 – Parte I. **Clube do Hardware.** Disponível em: <http://www.clubedohardware.com.br/artigos/707>. Acesso em: 23 de set. 2009.

VMWARE, INC. (2009a) **VMware.** Palo Alto, CA, Estados Unidos, 2009. Disponível em: <http://www.vmware.com>. Acesso em: 30 de ago. 2009.

VMWARE, INC. (2009b) **VMware Infrastructure sets world record for web server performance.** Palo Alto, CA, Estados Unidos, 17 fev. 2009. Disponível em: <http://www.vmware.com/company/news/releases/specweb2005.html>. Acesso em: 8 de nov. 2009.

WEICKER, Reinhold P. Dhrystone: a synthetic systems programming benchmark. **Communications of the ACM,** Los Angeles, CA, Estados Unidos, V. 27, N. 10, P. 1013 – 1030, out. 1984.

XEN. **Xen.org.** Disponível em: <http://www.xen.org>. Acesso em: 15 de set. 2009.

YORK, Richard. **Benchmarking in context: Dhrystone.** Cambridge, Inglaterra, 2002. Disponível em: <http://www.arm.com/pdfs/Dhrystone.pdf>. Acesso em: 12 de set. 2009.

ANEXO A – CONFIGURAÇÃO DO AMBIENTE PARA TESTES

I. Configuração do ambiente Linux para os testes

Instalação do sistema operacional SUSE Linux Enterprise Server 10 SP1 (X86)

- Mídia para instalação: CD-ROM, gravado a partir da imagem ISO disponível para *download* no site do fabricante
    - Endereço para download:

      http://download.novell.com/Download?buildid=2FNtOnmkx-w~
    - É necessário criar uma conta para efetuar a cópia dos arquivos
- Particionamento: automático (padrão), utilizando todo o disco disponível
- Software: pacotes “Server Base System” e “C/C++ Developer Tools”
- Informar a senha do usuário *root*
- *Firewall*: desligado
- Endereço IP: estático, dentro da faixa 192.168.2.xx
- *Novell Customer Center*: “*decide later*”
- *CA Management*: “*skip*”
- Criação de um novo usuário: informar nome e senha desejados
- *Hardware Config*: “*skip*”
- *Clone this system for AutoYAST*: não

Comandos para instalação do software MPICH (versão 1.2.7p1):

- Endereço para download dos códigos-fonte:

  ftp://ftp.mcs.anl.gov/pub/mpi/mpich-1.2.7p1.tar.gz
- Efetuar login no sistema como *root*
- Mudar para a pasta *home* do usuário e criar uma nova pasta chamada “mpich-1.2.7p1”

```
> cd ~
> mkdir mpich-1.2.7p1
```

- Mudar para a pasta *tmp* e descompactar o arquivo de instalação do MPICH

```
> cd /tmp
> tar -zxvpf ~/mpich.tar.gz
```

- Acessar a pasta recém-criada e exportar as variáveis de sessão abaixo:

```
> cd mpich-1.2.7p1
> export CC=gcc
> export CXX=g++
> export FC=gfortran
> export F77=gfortran
> export F90=gfortran
> export RSHCOMMAND=ssh
```

- Executar o script de configuração do MPICH

```
> ./configure -prefix=~/mpich-1.2.7p1 --enable-f77 --enable-
f90modules
```

Onde:

--prefix indica o diretório onde os arquivos serão instalados ao final do processo

--enable-f77 instrui o compilador a habilitar os componentes construídos em linguagem Fortran 77

--enable-f90-modules instrui o compilador a habilitar os módulos construídos em linguagem Fortran 90

- Gerar os executáveis e instalar os arquivos e documentação do MPICH nos locais determinados pelo script de configuração

```
> make; make install
```

- Exportar as seguintes variáveis de ambiente:

```
> export MPI="~/mpich-1.2.7p1"
> export PATH="${MPI}/bin:${PATH}"
> export LD_LIBRARY_PATH="${MPI}/lib:${LD_LIBRARY_PATH}"
```

Comandos para instalação do pacote de software ATLAS (versão 3.9.17):

- Endereço para download dos códigos-fonte:

  http://sourceforge.net/projects/math-atlas/files/Developer%20(unstable)/3.9.17/atlas3.9.17.tar.bz2/download

- Antes da instalação, é necessário verificar se o recurso de *CPU throttling* está ativado. Esse recurso é utilizado pelo subsistema de gerenciamento de energia do sistema operacional, com o intuito de economizar energia reduzindo o clock de funcionamento da CPU. Caso o ATLAS seja instalado nessas condições, o software apresentará comportamento instável.

  Para verificar o estado do *CPU throttling*, é necessário executar o comando abaixo:

```
> cpufreq-info
```

A saída desse comando será uma série de informações a respeito do estado atual da(s) CPU(s) do sistema. Entre essas informações, confirme que o "*governor*" ativado é "*performance*". Caso não seja, execute o comando:

```
> cpufreq-set -g performance
```

Esse comando instrui o sistema operacional a mudar o *scaling governor* da(s) CPU(s) presente(s) para máxima performance, ou seja, o processador passará a trabalhar a 100% de sua capacidade.

Obs.: no caso das máquinas virtuais, o processador estará obrigatoriamente sem o recurso de *CPU throttling*, devido ao fato do hardware apresentado pela ferramenta de virtualização ao SO hóspede não oferecer tal recurso.

- Acessar a pasta *home* e descompactar o arquivo de instalação do ATLAS:

```
> cd ~
> bunzip2 -c atlas.tar.bz2 | tar xfm -
```

- Uma pasta chamada ATLAS será então criada, com os arquivos de instalação do software. Criar uma nova pasta chamada "atlas-build" que servirá para armazenar os arquivos compilados do ATLAS para o sistema atual:

```
> mkdir atlas-build; cd atlas-build
```

- Executar o script de configuração do ATLAS:

```
> ~/ATLAS/configure -b 32 -D c -DPentiumCPS=2000
```

Onde:

`-b 32` indica que os arquivos devem ser compilados para execução em ambiente de 32 bits

`-D c -DPentiumCPS=<valor>` indica qual a frequência que deve ser considerada durante a medição dos tempos de *clock*. O parâmetro *<valor>* deve informar em MHz o *clock* da CPU do sistema - no caso, para um processador Sempron de 2.0 GHz, o valor correspondente é 2000.

- Gerar os executáveis, efetuar a verificação dos executáveis e testar o desempenho das bibliotecas recém-criadas:

```
> make build
> make check
> make time
```

Comandos para compilação e execução do *benchmark* HPL (versão 1.0a):

- Endereço para download dos códigos-fonte:

  http://www.netlib.org/benchmark/hpl/hpl.tgz

- Acessar a pasta *home*, descompactar os arquivos-fonte do HPL e acessar o diretório descompactado:

```
> cd ~
> gunzip hpl.tgz; tar -xvf hpl.tar
> cd hpl
```

- Copiar o arquivo modelo de configurações padrão (presente na subpasta "setup") que for mais semelhante ao hardware e ambiente a ser testado para um novo arquivo:

```
> cp setup/Make.Linux_ATHLON_CBLAS Make.DELL
```

- Editar o arquivo de configurações recém-criado:

```
> vi Make.DELL
```

- Alterar os parâmetros das linhas a seguir:

```
64    ARCH = DELL
...
84    MPdir = ~/mpich-1.2.7p1
...
95    LAdir = ~/atlas-build/lib
```

- Salvar o arquivo "Make.DELL" e fechar o editor de texto. Feito isso, gerar os executáveis utilizando o novo arquivo de configurações:

```
> make arch=DELL
```

- Será criada uma subpasta "bin/DELL", com os arquivos binários compilados para a arquitetura utilizada. Acessar a subpasta recém criada e editar o arquivo de configurações do *benchmark*, chamado "HPL.dat":

```
> cd bin/DELL; vi HPL.dat
```

- Alterar as linhas do arquivo conforme a sequência abaixo:

```
4     6
5     1
6     7296
7     1
8     128
9     0
10    1
11    1
12    1
13    16.0
14    1
15    2
16    1
17    4
18    1
19    2
20    1
21    1
22    1
23    1
24    1
25    1
26    2
27    64
28    0
29    0
30    1
31    8
32    ##### separador ######
33    0
34    1200 10000 30000
35    0
36    40 9 8 13 13 20 16 32 64
```

- Finalmente, executar o *benchmark* HPL, com o comando abaixo (dependendo do hardware testado, pode levar quinze minutos ou mais para execução):

```
> mpirun -np 1 xhpl
```

Comandos para compilação e execução do *benchmark* Bonnie++ (versão 1.03e):

- Endereço para download dos códigos-fonte:

  http://www.coker.com.au/bonnie++/bonnie++-1.03e.tgz

- Criar uma nova pasta "temp" no diretório raiz:

  ```
  > mkdir /temp
  ```

- Acessar a pasta *home* e descompactar os arquivos-fonte do Bonnie++:

  ```
  > cd ~; tar -xvf bonnie.tgz; mv bonnie++-* bonnie++
  > cd bonnie++
  ```

- Executar a compilação dos arquivos-fonte no sistema:

  ```
  > make
  ```

- Executar o *benchmark* com o comando abaixo:

  ```
  > ./bonnie++ -u root -d /temp -m nome -s 1024 -r 512 -x 5 -q
  ```

  Onde:

  `-u root` informa qual usuário o benchmark deve utilizar para criação do arquivo temporário, por questões de permissão de leitura/gravação

  `-d /temp` indica o diretório temporário para criação do arquivo de testes

  `-m nome` informa o nome da máquina testada (serve como rótulo para os resultados atingidos)

  `-s 1024` fornece o tamanho do arquivo temporário a ser utilizado (normalmente, utiliza-se o dobro da memória RAM instalada)

  `-r 512` fornece o tamanho da memória RAM instalada no computador

  `-x 5` indica o número de repetições do teste

  `-q` configura o *benchmark* para execução silenciosa, ou seja, somente os números finais do teste são exibidos na saída padrão

  Durante o desenvolvimento deste trabalho, todos os testes tiveram a saída redirecionada a um arquivo, utilizando o comando **>>** do Linux.

Comandos para compilação e execução do *benchmark* Netperf (versão 2.4.5):

- Endereço para download dos códigos-fonte:

  http://www.netperf.org/netperf/DownloadNetperf.html

- Nos dois computadores do teste, acessar a pasta *home* e descompactar os arquivos-fonte do Netperf:

```
> cd ~
> tar -zxvf netperf.tar.gz; mv netperf-* netperf
> cd netperf
```

- Executar a configuração do arquivo "makefile" e logo após a compilação dos arquivos-fonte no sistema, seguido da verificação dos arquivos gerados e instalação dos mesmos no diretório-padrão do programa:

```
> ./configure
> make; make check; make install
> cd /usr/local/bin
```

- No computador que irá trabalhar como servidor do Netperf, executar o processo servidor com o comando abaixo:

```
> ./netserver -d -p 12345
```

Onde:

-d habilita o modo verboso

-p <porta> indica a porta de rede onde o processo deve receber os dados (no caso, 12345)

- No computador que deve sofrer as medições, executar o processo cliente com o comando abaixo:

```
> ./netperf -H <ip-do-servidor> -p 12345 -l 120
```

Onde:

-H <ip-do-servidor> indica o endereço IP do computador que está executando o processo servidor do Netperf

-p 12345 indica a porta de rede onde o processo servidor Netperf está recebendo os dados

-l 120 informa o tempo em segundos que o teste deve ser executado (neste caso, 120 segundos; o padrão é 10 segundos).

Comandos para compilação e execução do *benchmark* STREAM (versão 5.9):

- Endereço para download dos códigos-fonte:

http://www.cs.virginia.edu/stream/FTP/Code/stream.c

- Acessar a pasta *home* e compilar o código-fonte do STREAM:

```
> cd ~
> gcc -O stream.c -o stream
```

Onde:

`-O stream.c` informa o arquivo que deve ser compilado

`-o stream` indica o nome do arquivo executável que deverá ser gerado após a compilação

- Executar o *benchmark* com o comando abaixo:

```
> ./stream
```

Cada execução do STREAM irá retornar as medições de desempenho do acesso à memória em quatro operações diferentes.

Comandos para compilação e execução do *benchmark* HINT (versão 1.0):

- Endereço para download dos códigos-fonte:

http://hint.byu.edu/pub/HINT/source/serial/unix/serial_hint.tar.gz

- Acessar a pasta *home* e descompactar o código-fonte do HINT:

```
> cd ~
> tar -zxvf serial_hint.tar.gz
> cd unix
```

- Editar o arquivo "*Makefile*":

```
> vi Makefile
```

- Alterar o arquivo "*Makefile*" na linha a seguir:

```
48   CC      = gcc
```

- Salvar arquivo e sair do editor. Efetuar a compilação dos códigos-fonte:

```
> make
```

- Executar o *benchmark* utilizando dados inteiros:

```
> ./INT
```

II. Instalação do software Sun VirtualBox (versão 3.0.10/54097):

- Mídia para instalação: arquivo de instalação no formato RPM, copiado através de *download* do site:

http://download.virtualbox.org/virtualbox/3.0.10/VirtualBox-3.0.10_54097_sles10.1-1.i586.rpm

Requisito mínimo de sistema: Linux com *X Window System*

Dependências: pacotes **`pam-devel`** e **`libsdl-1.2so.0`**

- Alterar o *scaling governor* da CPU para o modo "performance":

```
> cpufreq-set -g performance
```

- Acessar a pasta *home* e executar o instalador RPM do VirtualBox:

```
> cd ~
> rpm -i VirtualBox.rpm
```

- Iniciar o VirtualBox:

```
> VirtualBox
```

- Criar uma nova máquina virtual com as seguintes configurações:
  - Sistema operacional: Linux / versão: openSUSE
  - RAM: 512 MB
  - Hard Disk: criar novo disco virtual de tamanho fixo de 8 GB
  - CD/DVD: montar *drive* físico
  - Floppy: não
  - Áudio: não
  - Rede: conectar adaptador virtual de rede "eth0" à rede virtual "*bridged*", com endereço IP na faixa 192.168.2.XX
  - Porta serial: não
  - USB: controlador EHCI 2.0

Após a criação da VM, instalar o ambiente Linux e executar *benchmarks* como descrito na seção I deste Anexo.

III. Instalação do software VMware Server (versão 2.0.2/203138):

- Mídia para instalação: arquivo de instalação no formato RPM, copiado através de *download* do site abaixo (é necessário criar uma conta para acessar o arquivo):

  http://www.vmware.com/download/server/getserver.html

  Requisito mínimo de sistema: Linux com *X Window System*

  Requisito mínimo para acesso à ferramenta de gerenciamento: navegador Microsoft Internet Explorer 6.0 ou mais recente ou Mozilla Firefox 3.0 ou mais recente

- Alterar o *scaling governor* da CPU para o modo "performance":

```
> cpufreq-set -g performance
```

- Acessar a pasta *home* e executar o instalador RPM do VMware Server:

```
> cd ~
```

```
> rpm –i VMwareServer.rpm
```

- Executar o script de configuração do programa para o sistema operacional instalado:

  ```
  > /usr/Bin/vmware-config.pl
  ```

  Seguir os passos na tela, respondendo às questões e inserindo o número serial de registro fornecido pelo fabricante.

- Acessar via *browser* o endereço abaixo:

  https://ip-do-equipamento:8333

  Na tela de login que surge, informar como usuário *root* e a respectiva senha.

- Criar uma nova máquina virtual: Selecionar a opção *Create VM*
  - Nome: SUSE
  - Sistema operacional: Linux / versão: SUSE Linux Enterprise Server 10 SP1
  - RAM: 512 MB
  - *Virtual Disk*: criar novo disco virtual de tamanho fixo de 8 GB
    - Selecionar a opção "*allocate all disk space now*"
  - Adaptador de rede: selecionar a rede "*bridged*"
  - CD/DVD: montar *drive* físico
  - Floppy: não
  - USB: sim

Após a criação da VM, instalar o ambiente Linux e executar *benchmarks* como descrito na seção I deste Anexo.

IV. Instalação do software Citrix XenServer (versão 5.5.0):

- Mídia para instalação: para o XenServer, dois CDs, criados a partir de imagens ISO obtidas através de *download* do site do fabricante; para o XenCenter, um arquivo no formato MSI para instalação em ambiente Windows, obtido através de *download* do site do fabricante
  - Instalador do XenServer:

    http://downloadns.citrix.com.edgesuite.net/akdlm/4210/FREE_XenServer-5.5.0-install-cd.iso
  - *Linux Guest Support CD:*

    http://downloadns.citrix.com.edgesuite.net/akdlm/4211/FREE_XenServer-5.5.0-linux-cd.iso

- Instalador do XenCenter:

  http://downloadns.citrix.com.edgesuite.net/akdlm/4212/FREE_XenServer-5.5.0-XenCenter.msi

Requisito mínimo de sistema: hardware X64 compatível

Requisito mínimo para acesso à ferramenta de gerenciamento XenCenter: máquina com Microsoft Windows 2000 SP4 ou mais recente, *.NET Framework* 2.0 instalado e conectada à mesma rede do servidor.

No servidor:

- Inserir o primeiro CD no *drive* do servidor e inicializar o equipamento, selecionando o CD-ROM como dispositivo para *boot.*
- Seguir os passos na tela, conforme solicitado pelo instalador do XenServer.

No computador com Windows:

- Efetuar a instalação do XenCenter a partir do arquivo de instalação do programa.
- Abrir o XenCenter e selecionar a opção "*Connect to Server*".
- Na tela que surge, informar o endereço IP do servidor com XenServer, o usuário e senha de gerenciamento (configurados durante a instalação do servidor) e clicar em OK.
- Após a sincronização do servidor com o XenCenter, clicar sobre o servidor listado na ferramenta e depois em "*New VM*".
  - *Template*: SUSE Linux Enterprise Server 10 SP1
  - Nome: SUSE
  - *CD/DVD Location: physical DVD drive (drive 0)*
  - *Initial RAM*: 512 MB
  - *VCPUs*: 1
  - *Virtual disks:* criar novo disco de 8 GB em *local storage*
  - *Network interfaces: auto-generate*

Após a criação da VM, instalar o ambiente Linux e executar *benchmarks* como descrito na seção I deste Anexo.

ANEXO B – CONFIGURAÇÃO DO AMBIENTE XENSERVER PARA MIGRAÇÃO DE MÁQUINAS VIRTUAIS

I. Configuração do sistema XenLinux no servidor que irá oferecer o volume compartilhado

Pré-requisitos: instalação do software Citrix XenServer em pelo menos um servidor da rede. Este servidor irá disponibilizar um volume compartilhado através de NFS para os servidores restantes.

Criação e compartilhamento de um novo volume através de NFS:

- Efetuar *login* com o usuário *root* do servidor:
  - Na tela de *status* do servidor desejado, selecionar a opção "*Local Command Shell*"
  - Inserir usuário *root* e a respectiva senha

- Eliminar o volume padrão "*Local storage*" criado pelo instalador do XenServer:
  - Verificar a listagem de volumes de armazenamento (*storage repositories*, ou SRs) do tipo LVM disponíveis nos servidores:

    > xe sr-list type=lvm

    Anote o código UUID do volume.

  - Verificar a listagem de *physical block devices* (PBDs) montados sobre o SR "*Local storage*":

    > xe pbd-list sr-uuid=<SR-UUID>

    Onde **<SR-UUID>** é o código UUID do SR "*Local storage*", exibido no comando anterior.

    Anote o código UUID do PBD exibido com este comando.

  - Desconectar o PBD montados sobre o SR "*Local storage*":

    > xe pbd-unplug uuid=<PBD-UUID>

    Onde **<PBD-UUID>** é o código UUID do PBD exibido no comando anterior.

    Neste momento, o SR "*Local storage*" deve aparecer no XenCenter com um ícone de alerta, indicando que há problemas com esse recurso. Não efetuar o reparo do SR no XenCenter; ignorar tal aviso.

- Finalmente, eliminar o SR "*Local storage*":

```
> xe sr-destroy uuid=<SR-UUID>
```

Onde **<SR-UUID>** é o código UUID do SR "*Local storage*", exibido anteriormente.

Neste momento, o SR "*Local storage*" deverá desaparecer da listagem do XenCenter, que é o objetivo desejado.

- Criar uma nova área de armazenamento com sistema de arquivos EXT3 e montá-la em um novo diretório na raiz do sistema (esse novo diretório deve obrigatoriamente receber permissões de execução para todos os usuários do sistema):

```
> mkfs –t ext3 /dev/sda3
> mkdir /discos
> mount /dev/sda3 /discos
> chmod 777 /discos
```

- Compartilhar o diretório recém-criado através de NFS:
  - Editar o arquivo "**/etc/exports**":

```
> vi /etc/exports
```

Inserir uma nova linha conforme abaixo:

```
/discos *(rw,async)
```

Onde:

/discos informa o caminho do diretório a compartilhar

***(rw)** concede permissões de leitura e gravação a todos os usuários da rede

  - É necessário configurar o firewall do XenLinux para permitir conexões às portas de comunicação do NFS. Neste trabalho, o firewall foi totalmente desabilitado com o comando abaixo:

```
> service iptables stop
```

  - Por padrão, o processo "**portmapper**" do XenLinux é iniciado com o parâmetro "**–l**", que indica que esse processo irá responder somente a requisições da própria máquina. Portanto, é necessário reiniciar esse

processo sem esse parâmetro, para que as outras máquinas da rede possam comunicar com o “portmapper”:

```
> kill <PID do portmap>
> portmap
```

- Parar e iniciar novamente o serviço NFS do XenLinux:

```
> service nfs stop
> service nfs start
```

- Exportar a lista de diretórios compartilhados:

```
> exportfs
```

- Verificar a lista de processos RPC que estão sendo executados:

```
> rpcinfo –p <IP do servidor>
```

Caso os processos **mountd**, **nfsd** e **statd** não estejam sendo executados, é necessário chamá-los com os comandos:

```
> rpc.mountd
> rpc.nfsd
> rpc.statd
```

Caso todos os comandos tenham sido executados corretamente, o volume NFS estará pronto para ser montado no XenCenter.

II. Criação de um novo Storage Repository

Através da interface gráfica do XenCenter:

- Abrir a ferramenta XenCenter e conectar-se ao *pool* de servidores
- Selecionar o menu “*Storage* / *New Storage Repository*”
  - Na janela que aparece, selecionar a opção “*NFS VHD*” e “*Next*”
  - No campo “*Name*“, informar o nome desejado para o novo SR
  - No campo “*Share Name*”, informar o caminho NFS para o diretório compartilhado no servidor, conforme abaixo:

    192.168.2.3:/discos

    Onde:

    192.168.2.3 é o endereço IP do servidor com o diretório compartilhado

    /discos informa o caminho do diretório compartilhado

- Clicar em "*Finish*"

  Caso não haja problemas de configuração com o diretório compartilhado, o novo SR irá aparecer na listagem de recursos do *pool*, dentro da janela do XenCenter.

Através da linha de comandos do XenLinux:

- Efetuar *login* com o usuário *root* do servidor:
  - Na tela de *status* do servidor desejado, selecionar a opção "*Local Command Shell*"
  - Inserir usuário *root* e a respectiva senha

- Criar o novo SR com os comandos abaixo:

```
> xe sr-create content-type=user name-label=<NOME_DO_SR> \
  shared=true type=nfs device-config:server=<IP_SERVIDOR> \
  device-config:serverpath=<DIRETORIO>
```

  Onde:

  `content-type=user` indica que o SR foi criado manualmente

  `name-label=<NOME_DO_SR>` informa o nome do novo SR

  `shared=true` informa que o novo SR será compartilhado na rede

  `type=nfs` indica que o novo SR será para um diretório acessível via NFS

  `device-config:server=<IP_SERVIDOR>` indica o endereço IP do servidor que compartilha o diretório desejado

  `device-config:serverpath=<DIRETORIO>` indica o caminho do diretório compartilhado dentro do servidor

  Caso o comando seja executado com sucesso, o novo SR irá aparecer no XenCenter, na listagem de recursos do *pool*.