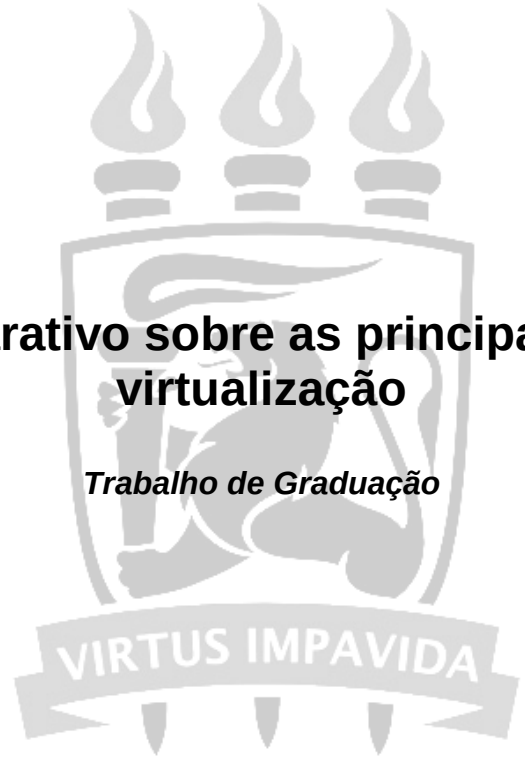
UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
2006.2



Um estudo comparativo sobre as principais ferramentas de virtualização

Trabalho de Graduação

Aluno: Marcos Tadeu de Andrade (mta@cin.ufpe.br)
Orientador: André Santos (alms@cin.ufpe.br)

Índice

Índice de Figuras

Aos meus pais.

Resumo

Máquinas Virtuais são abstrações de hardware criadas pelos chamados monitores de máquinas virtuais. O uso de máquinas virtuais tem crescido em ambientes de administração de sistemas, adicionando recursos extras de hardware onde nem sempre é possível a compra e utilização de novos equipamentos. Este trabalho tem como objetivo elucidar os principais conceitos no contexto de máquinas virtuais e analisar as principais técnicas de virtualização através de produtos disponíveis no mercado, comparando-os em relação a seu desempenho em diversas tarefas.

Palavras-chaves: Sistemas Operacionais, Virtualização, emulação, paravirtualização, Xen, VMware, *benchmarking*.

Agradecimentos

A Deus. Por tudo que tem feito por mim, durante toda minha vida.

Aos meus pais, João e Lourdes, por seu amor e carinho, e por me ensinarem o valor do estudo e me incentivarem em todos os momentos a perseguir meus objetivos, ainda que sonhadores demais.

Aos meus irmãos, Jairo, Leandro e Viviane, por sua amizade (ainda que com as tradicionais brigas...) e por sempre estarem presentes em momentos especiais da vida.

À minha avó, Quitéria, por todo amor e dedicação desprendidos comigo, e pelo apoio, inclusive financeiro, nos momentos de maior dificuldade.

Aos meus amigos em minha cidade natal, Caruaru, pelo tempo juntos conversando e falando bobagem.

Aos meus amigos, antes virtuais, agora mais que reais (“amigos de infância...”), Jeisa, Jackeline, Jeasi, Jonathas, Ellen, Marcia, Camila, Guaíra, Guaciara e tantos outros que vieram por meio deles, me ajudando em tudo quanto precisei.

À eterna equipe “dos caras”, Bruno “bpe”, Marcio “mpl”, Lauro Moura e Adelmário, pelos alucinantes períodos de stress pré-entrega de projetos, e pelos projetos feitos em tempo mais que mínimo que conseguimos entregar a tempo.

Aos não mais colegas, mas sim amigos, que encontrei na faculdade, Allan, Aninha, Bengt, Bruno Ferreira, Chico “Bala”, Carlinhos, Carlos, Carol, Mano “psicopata”, Gera, Joabe, Lais, Nancy, Pablo, Martinelli, René, Thierry, Arthur, Rodrigo, Paulo César “PC” e todos os outros que conheci e que não coloquei aqui.

Aos meus colegas de trabalho, Juliana, Nadja Lins, Gilce, Jorge, Cláudio e Ana, e minha chefe, Marlice “Mel” Novais, pela compreensão pelo tempo dispendido na elaboração deste documento e pelos conselhos muito úteis.

Ao meu orientador, André Santos, pelo tempo disponibilizado entre suas tantas atividades, enviando material para a elaboração do trabalho e sugestões de melhorias do mesmo.

Aos demais professores do Centro de Informática, em especial Fernando Fonseca e Sílvio Meira.

A todos os outros que passaram por minha vida, colaborando direta ou indiretamente na realização deste trabalho.

A todos esses, meus mais sinceros agradecimentos.

“… não tenho medo nem de chuvas tempestivas nem de grandes ventanias soltas…”.

Clarice Lispector - Escritora Brasileira (1920-1977)

1 Introdução

Nos últimos anos, a capacidade de processamento dos computadores tem aumentado bastante. Entretanto, toda a capacidade adquirida não tem sido totalmente aproveitada. Temos situações onde durante a maioria do tempo que poderia ser usado para o processamento de aplicações não é utilizado como desejaríamos que fosse. Uma solução é a utilização de máquinas virtuais, que estão cada vez mais sendo adotadas por empresas em geral e por empresas de pequeno e médio porte [Eweek, 2007]. A vantagem de se maximizar o tempo de processamento é permitir mais processos executando simultaneamente, tornando os equipamentos já adquiridos mais produtivos (ao menos em teoria).

A utilização de máquinas virtuais não é algo recente. Já na década de 1960 um grupo formado por pesquisadores da IBM e do MIT foi responsável pelo projeto M44/44X. Logo depois, a IBM começou a desenvolver uma série de sistemas operacionais compatíveis com virtualização, como o VM/370 ([Creasy, 1981]), com o fim de prover um acesso concorrente e interativo a um servidor *mainframe*. Desde então várias soluções, tanto comerciais quanto acadêmicas, foram propostas. Atualmente, a maioria dessas soluções utiliza as técnicas conhecidas como virtualização completa, enquanto um pequeno número aborda as técnicas de para-virtualização.

1.1. Objetivos do trabalho

O presente trabalho tem como objetivo elucidar os principais conceitos envolvidos em virtualização e analisar o comportamento e a performance de alguns monitores de máquinas virtuais, nos principais produtos encontrados no mercado.

1.2. Estrutura do trabalho

Este trabalho está dividido em 5 seções. Na primeira, vemos os principais conceitos envolvidos no tema (virtualização). Na seção seguinte temos as configurações de *hardware* e *software* utilizadas no trabalho. A terceira seção detalha as ferramentas utilizadas no processo de avaliação da performance de cada um dos produtos. A quarta seção mostra os resultados dos testes realizados e comentários sobre eles. Na quinta e última seção, temos uma conclusão geral, com uma análise dos resultados, além das dificuldades encontradas e sugestões para trabalhos futuros, tendo este trabalho como ponto de partida.

2 Virtualização, emulação e outros conceitos

2.1 Virtualização

Virtualização pode ser definida como uma “técnica que combina ou divide recursos computacionais para prover um ou mais ambientes operacionais de execução” [NandaChiueh, 2005]. Os ambientes criados através dessa técnica são chamados máquinas virtuais.

Dentre os conceitos envolvidos no estudo de máquinas virtuais, o de um monitor (também chamado de *hypervisor*, VMM, ou Virtual Machine Monitor) é um dos principais. O monitor é uma camada de software inserida entre o sistema visitante (*guest system*) e o hardware onde o sistema visitante executa [XenSource, 2007]. Essa camada faz uma interface entre os possíveis sistemas visitantes (virtuais) e o hardware que é compartilhado por eles. Ela é responsável por gerenciar todas as estruturas de hardware, como MMU, dispositivos de E/S, controladores DMA, criando um ambiente completo (máquina virtual), onde os sistemas visitantes executam.

Segundo [King, 2003], VMMs podem ser classificados de várias maneiras. Uma delas refere-se a quão perto a interface que eles provêem está do hardware subjacente. VMMs como o VMware ESX Server e emuladores como o Bochs apresentam uma abstração de hardware idêntica ao hardware subjacente. Para chegar a um nível de abstração tão alto, alguns monitores inserem no SO hóspede(*host OS*) *device drivers* que serão usados pelo SO visitante (*guest OS*) através do VMM. Já outros VMMs apresentam uma interface diferente do hardware subjacente. A máquina virtual Java tem uma arquitetura totalmente independente do hardware sobre o qual executa.

Outra classificação de VMMs se refere à plataforma sobre a qual eles executam [Goldberg, 1973]. Segundo essa classificação, as máquinas virtuais podem ser de 2 tipos [King, 2003]:

- Tipo I - O VMM é implementado diretamente sobre o hardware físico subjacente. Os VMMs Xen e VMware ESX Server são exemplos desse tipo de máquinas virtuais.

- Tipo II - O VMM é implementado completamente sobre o sistema operacional *host*. Exemplos desse tipo de máquinas virtuais são o VMware Server o VirtualPC.

As figuras 1 e 2 mostram esquemas das arquiteturas de Tipo I e Tipo II.

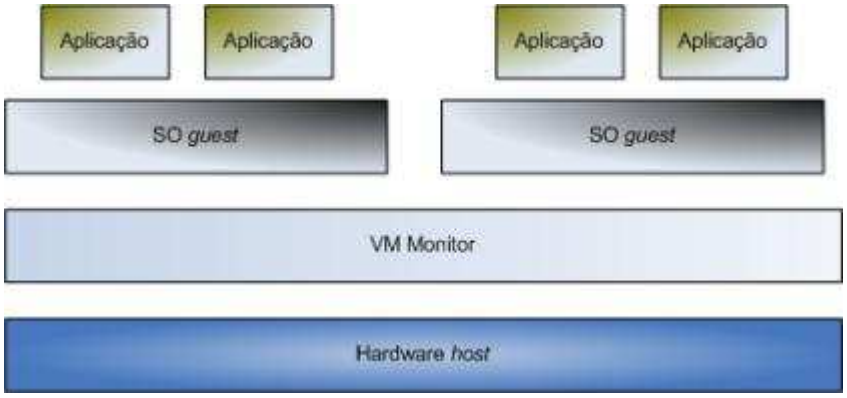


Figura 1 - Arquitetura tipo I

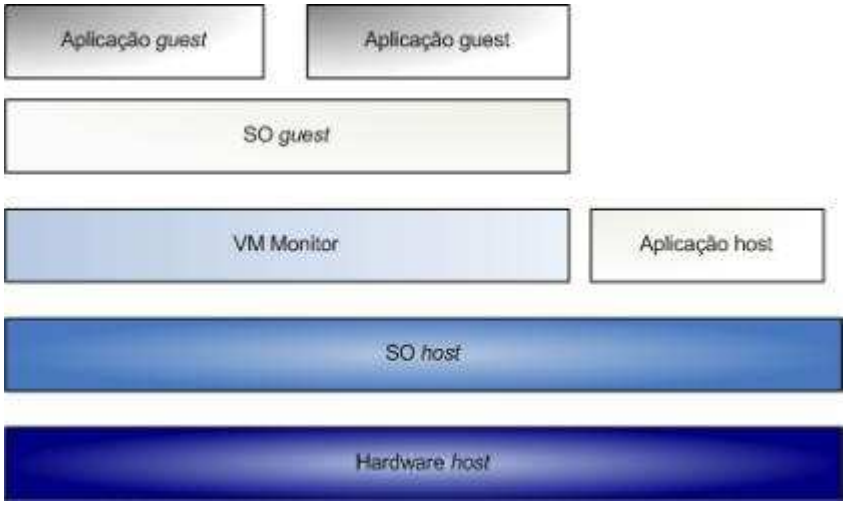


Figura 2 - Arquitetura tipo II

Algumas modificações podem ser inseridas nas arquiteturas descritas anteriormente com o objetivo de aumentar o desempenho das máquinas virtuais. Essas modificações geram arquiteturas híbridas. Exemplos dessas arquiteturas podem ser encontrados em [Laureano, 2006].

2.2 Emuladores

Um emulador é um sistema que simula toda uma arquitetura computacional, de forma que todas as instruções do sistema operacional (SO) emulado são traduzidas e executadas no SO original. Comumente, emuladores são usados para testar o funcionamento de programas escritos para arquiteturas diferentes daquelas em que o emulador executa.

A diferença entre emuladores é máquinas virtuais, está basicamente no nível de abstração. Enquanto em um emulador toda as instruções são convencionalmente traduzidas e interpretadas, em máquinas virtuais as instruções são executadasno modo mais nativo possível.

Emuladores podem ser classificados de várias formas. Quanto à sua natureza de uso, podem ser classificados como: ([Laureano, 2006]):

- Emuladores de processador.
- Emuladores de sistemas operacionais.
- Emuladores de uma plataforma de hardware específica.
- Emuladores de videogames (consoles).

Como exemplo de emuladores de processadores temos o *Bochs* [Bochs, 2007], que emula processadores x86. Sua performance não é muito alta, devido ao fato de que as instruções são interpretadas e executadas uma por uma. Sistemas como o *Bochs* são mais usados como simuladores, tendo mais utilidade em projetos de software *low level* (como sistemas operacionais) ou de simulação de novas arquiteturas de hardware [Bjerke, 2005]. Uma abordagem que não se utiliza da tradução "instrução-por-instrução" é a tradução de blocos de instruções para posterior execução nativa. Desse modo, para blocos de código já traduzidos não é necessário uma retradução. Dentre os emuladores que se valem dessa técnica temos os emuladores de sistemas operacionais Microsoft Virtual PC [Microsoft, 2007], e QEMU [QEMU, 2005]. Além desses, temos os emuladores de plataformas que não mais existem, como o UAE Amiga Emulator [UAE, 2005]. Também são

bastante conhecidos os emuladores de consoles (videogames), como o ePSXe [ePSXe, 2003], o SNES9x [SNES9x, 1999] e o ZSNES [ZSNES, 2001]. Podemos citar ainda o WINE [WINE, 1993], que é um emulador de bibliotecas de usuário.

2.3 Técnicas de Virtualização

Existem várias técnicas usadas na virtualização. As principais são virtualização completa, paravirtualização e recompilação dinâmica. Essas técnicas são detalhadas nos subtópicos seguintes.

2.3.1 Virtualização completa

Na virtualização completa (*full virtualization*), toda uma infra-estrutura do hardware subjacente é virtualizada, de forma que não é necessário modificar o sistema operacional convidado para que o mesmo execute sobre o VMM. Podem ocorrer, entretanto, penalidades em relação ao desempenho da máquina virtual, uma vez que já que o hardware é virtualizado, as instruções devem ser interpretadas pelo VMM. Uma desvantagem dessa técnica na arquitetura x86 é que a mesma não foi projetada tendo em vista a virtualização, mas sim teve uma evolução a partir de versões anteriores. Assim, algumas instruções privilegiadas que executam em modos diferentes (modo usuário ou modo supervisor) geram resultados diferentes dependendo do modo em que executam [XenSource07a]. Uma solução abordada pelo VMware ESX Server é verificar um pedaço do código que está sendo executado na máquina virtual e modificar (*patch*) as instruções que poderiam resultar em erros. Essa solução é conhecida como *binary patching*.

2.3.2 Paravirtualização

A paravirtualização é um método que consiste em apresentar ao SO que está sendo emulado uma arquitetura virtual que é similar, mas não idêntica à arquitetura física real [XenSource, 2007]. Essa solução aumenta a performance das máquinas virtuais que a utilizam [Barham. 2003]. Entretanto, são necessárias modificações nos sistemas operacionais convidados, que executam na atual arquitetura x86. Ainda assim, as mudanças necessárias nos sistemas convidados devem ser passíveis de implementação, o que já é implementado nas versões mais recentes do Linux (versão 2.6). Recentemente, a Microsoft anunciou uma parceria com a companhia XenSource para que versões futuras do Windows Server, através de virtualização, possam executar distribuições Linux que utilizam o Xen como VMM [Microsoft, 2006]. Isso mostra que a modificação do sistema operacional convidado é possível de ser feita com um esforço não tão grande a ponto de impactar substancialmente na evolução de SOs já consolidados no mercado.

2.3.3 Recompilação Dinâmica

Essa técnica, que é conhecida também pelo nome de tradução dinâmica (*dynamic translation*), consiste em traduzir durante a execução de um programa as instruções de um formato para outro. Uma aplicação da técnica é vista em compiladores JIT (*just-in-time*), que traduzem de uma linguagem *bytecode* para código nativo da CPU onde o compilador executa. A recompilação (ou tradução) é feita em 7 passos, descritos em [Tijms, 2000]. Em um primeiro momento, o código binário é escaneado para que seja identificado uma seqüência de bits correspondente à seção de código do programa em execução. Logo após esse passo, os bits agrupados anteriormente são divididos em instruções, juntamente com os parâmetros delas. Então, as instruções são transformadas para uma representação mais próxima da máquina nativa. Um código em uma linguagem de alto nível é gerado a partir da representação anterior, código esse que é compilado e reescrito na linguagem nativa. Assim, temos uma seqüência de bits agora

executável no hardware nativo.

Emuladores como o QEMU utilizam essa técnica para aumentar seu desempenho [Bellard, 2005]. O VMware Workstation também utiliza essa técnica, recompilando apenas parte do código, uma vez que boa parte dele pode executar nativamente (a arquitetura de hardware subjacente é a mesma da máquina virtual). No VMware Workstation apenas instruções que não podem ser executadas diretamente são recompiladas [Laureano, 2006].

Um projeto da antiga DEC (DIGITAL FX! 32) [Chernoff and Hookway, 1997] utilizava um misto de emulação e tradução binária para executar aplicações 32-bit que executavam no sistema operacional Windows NT 4.0 em computadores *Alpha*. Enquanto as aplicações executavam, o emulador capturava um perfil de execução, que, por sua vez, era usado pelo tradutor binário para traduzir as partes das aplicações que haviam executado em código nativo do processador *Alpha*.

Outro exemplo de recompilação dinâmica ocorre em interpretadores de linguagens, como no interpretador Sun "Hotspot" da linguagem Java [Sun, 1999]. As instruções geradas para a máquina virtual e armazenadas nos *bytecodes* das classes Java são traduzidas e executadas no hardware subjacente.

2.4 Tecnologias facilitadoras do trabalho de virtualização

Conforme citado anteriormente (seção 2.3.1), a arquitetura x86 não foi projetada tendo em vista a virtualização. Assim, alguns fabricantes (AMD e Intel) propuseram novas arquiteturas que dão suporte a essa tecnologia. Ambos os fabricantes também contribuíram em questão de código no desenvolvimento do VMM Xen.

Tradicionalmente, os processadores implementam 4 níveis (ou *rings*, na terminologia usada para execução de processos) de proteção [Jones, 2007]. Quanto mais alto o nível, menos privilégios para a execução de instruções são concedidos. O nível 0 (*ring-0*) é geralmente usado pelo sistema operacional, níveis 1 e 2 pelos processos do sistema operacional e o nível 3 pelas aplicações.

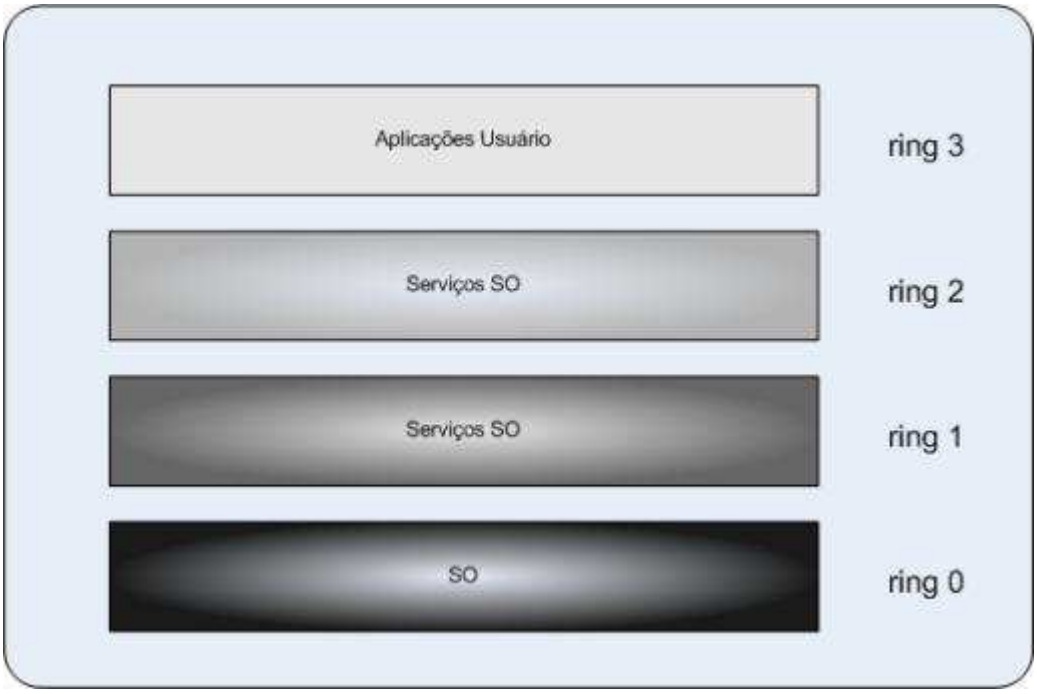


Figura 3 - Anéis(*rings*) de proteção da arquitetura x86 atual

A solução VT-x [Intel, 2007] (para processadores da família x86, 32 bits) proposta pela Intel consiste em fornecer duas novas formas para operações da CPU [Leung, 2006]: VMX *root operation* e VMX *non-root operation*. VMMs executam operações do tipo *root* no *ring-0*, enquanto que as máquinas virtuais executam

operações do tipo *non-root*, em níveis inferiores de prioridade (*ring-1* ou *ring-2*). Processadores equipados com essa tecnologia implementam diretamente em hardware métodos que seriam usados pelo VMM para a execução de instruções privilegiadas oriundas dos sistemas operacionais virtualizados (*hardware assist*).

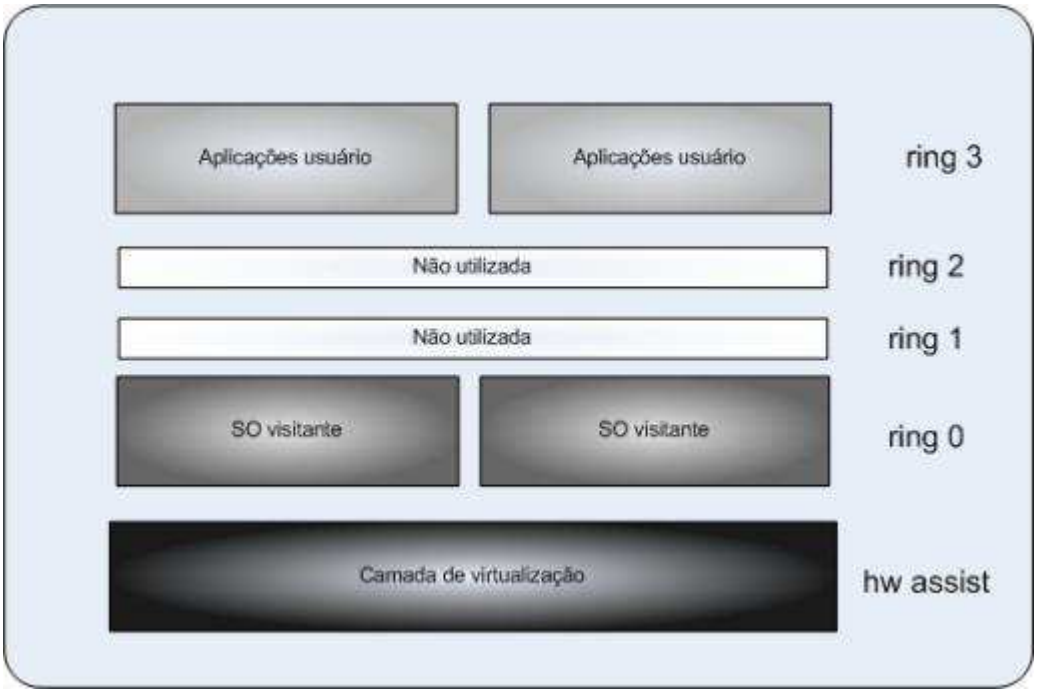


Figura 4 - Arquitetura Intel VT-x

A AMD adotou uma postura semelhante à da Intel. Na arquitetura AMD-V [AMD, 2007] (inicialmente AMD Pacifica), o processador mantém um bloco de controle para os sistemas convidados que é salvo quando instruções especiais são executadas. O VMM pode customizar os privilégios de execução para cada um dos sistemas convidados.

2.5 Propriedades de VMMs

Algumas propriedades dos VMMs devem ser observadas para o funcionamento ideal de uma máquina virtual.

- Isolamento - um processo em execução em uma máquina virtual não deve interferir no funcionamento de outro processo em execução, tanto no

monitor quanto em outra máquina virtual. Também é necessário que o VMM gerencie se o funcionamento de um processo numa máquina virtual está comprometendo o desempenho de outras máquinas virtuais.

- Inspeção - o VMM deve ser capaz de ter acesso e controle a todas as informações dos processos em execução na máquina virtual, como memória, estado da CPU e outros.

- Interposição - instruções podem ser inseridas pelo VMM em operações da máquina virtual. Um exemplo é quando uma máquina virtual tenta executar uma instrução privilegiada. Instruções do tipo "espera" podem ser inseridas na máquina virtual enquanto o VMM processa as instruções privilegiadas.

- Eficiência - instruções "inofensivas" podem ser executadas diretamente no hardware, por não interferir nas outras máquinas virtuais ou no VMM.

- Gerenciabilidade - possibilidade de gerenciar uma máquina virtual independente de outra VM, dado que são entidades distintas.

- Compatibilidade do software - todo software escrito para executar sobre uma plataforma deve ser passível de execução em uma VM que virtualiza essa plataforma.

- Encapsulamento - o VMM pode controlar totalmente processos em execução em VMs, podendo gerar *"checkpoints"* do sistema para possíveis recuperações pós-falhas.

- Desempenho - apesar de inserir uma camada extra de software a um sistema pode comprometer o desempenho de um SO, mas os possíveis benefícios de uso de máquinas virtuais compensam os prejuízos.

2.6 Uso e aplicações de máquinas virtuais

As máquinas virtuais podem ser usadas em muitos cenários práticos [NandaChiueh05]. Alguns deles são:

- Ensino e aprendizagem - máquinas virtuais podem ser usadas no ensino de sistemas operacionais, bem como na aprendizagem do funcionamento deles. Uma VM pode ser facilmente substituída por outra VM caso ocorra algum erro durante o uso da mesma.

- *Honeypots* e *honeynets* - um *honeypot* é um sistema colocado em uma rede com o objetivo de ser comprometido. Uma *honeynet* é uma rede formada por vários *honeypots*. A grande vantagem de se utilizar VMs em uma *honeynet* é não comprometer a rede real, sendo os ataques confinados às *honeynets*. Assim, quando um dos *honeypots* é comprometido, pode ser substituído por outro de forma mais fácil.

- Consolidação de servidores - consolidar *workloads* de máquinas sub-utilizadas em poucas máquinas, economizando hardware, gerenciamento e administração da infraestrutura.

- Consolidação de aplicações - aplicações legadas que necessitam executar em um novo hardware, diferente do hardware para qual foram projetadas. Virtualizando o hardware, essas aplicações podem continuar executando normalmente.

- *Sandboxing* - máquinas virtuais podem prover um ambiente seguro e isolado para a execução de aplicações não confiáveis, ou de fontes não seguras.

- Ambientes múltiplos de execução - a virtualização pode prover múltiplos ambientes de execução e aumentar a disponibilidade de recursos para as aplicações.

- Hardware virtual - pode ser possível, através da virtualização, prover

um hardware que não exista na máquina real, como drives SCSI virtuais, interfaces virtuais de rede e outros.

- Múltiplos SOs simultâneos - vários SOs podem executar simultaneamente, fazendo que uma gama maior de aplicações possa estar executando ao mesmo tempo.

- *Debugging* - a virtualização pode prover para um desenvolvedor de aplicações como device drivers ou mesmo um SO executar o software em um hardware com um controle total.

- Migração de software - facilita a migração de software e aumenta a mobilidade de softwares.

- *Appliances* - algum indivíduo ou organização pode disponibilizar sua aplicação escrita para um SO como um *appliance* (conjunto de softwares empacotados e prontos para executar), distribuindo toda a infraestutura necessária para que sua aplicação execute corretamente.

- Testing/QA - a virtualização torna possível produzir cenários de teste que são difíceis de produzir em máquinas reais e assim facilitam o teste de software.

3 Configuração do software e do hardware escolhidos para os testes

No presente trabalho, foi escolhido como sistema operacional de testes a versão 3.1 da distribuição Debian GNU/Linux. Para a escolha do sistema operacional, foram tomados como parâmetros a grande confiabilidade do Debian GNU/Linux (apesar de o mesmo não apresentar um conjunto de pacotes de software tão atualizado como o das outras distribuições GNU/Linux), a inclusão de templates para instalação nos VMMs utilizados (em especial, no XenExpress) e o utilitário de gerenciamento de pacotes ***apt-get***. O hardware utilizado é um servidor x86 de pequeno porte (PC), que são utilizados em pequenas e médias empresas (que, ao contrário de grandes organizações, não possuem recursos financeiros disponíveis para a aquisição de hardware dedicado a servidores - Sun, IBM e outros). Foi necessário também a utilização de um segundo equipamento, que atua como o cliente dos testes de rede descritos nas seção 3.1.3 e 3.1.6.

O hardware utilizado nos testes tem as seguintes especificações:

- Equipamento principal
  - Processador: Intel Pentium D 3.0 GHz, 1MB cachê
  - Placa de rede: Intel Pro/100 Ethernet
  - Memória total: 4GB
  - Disco: Samsung SP080ZN, 80GB, IDE
- Hardware auxiliar, utilizado nos testes de rede:
  - Processador: Intel Pentium 4 3.0 GHz, 2MB cache, 1 core
  - Placa de rede: Tigon3 10/100/1000BaseT Gigabit Ethernet
  - Memória total: 4GB
  - Disco: 250 GB, SATA

Os softwares utilizados nos testes são os seguintes:

- Sistema Operacional: GNU/Linux Debian 3.0
- Virtual Machine Monitors (VMMs)
    - VMware Server 1.0.2
    - XenExpress 3.1

Os softwares escolhidos levam em consideração os principais produtos disponíveis no mercado para uma solução de virtualização. Um deles (VMware Server) utiliza as técnicas de virtualização completa, enquanto o outro (XenExpress) utiliza paravirtualização. O VMware Server é uma opção para aqueles que planejam começar uma infra-estrutura de virtualização mas não possuem muitos recursos ou desejam apenas verificar, na prática, as vantagens da virtualização. O XenExpress é uma solução baseada no Xen e inclui uma versão modificada e reduzida do kernel do linux, executando somente os serviços essenciais do VMM (escalonador, drivers de E/S e outros). O XenExpress é um VMM de tipo II (ver seção 2.2), enquanto o VMware Server é um VMM do tipo I.

O XenExpress apresenta uma interface de gerenciamento simples que permite a visualização do status das máquinas virtuais(Xen Virtual Machines) e as características da máquina hospedeira (XenServer). Através da interface de gerenciamento, também é possível utilizar os serviços disponibilizados pelas interfaces gráficas dos sistemas hóspedes, fazendo com que usuários menos experientes não tenham a necessidade de configurar manualmente esses serviços.

O VMware Server fornece duas opções de ferramentas que permitem gerenciar as máquinas virtuais em execução. Temos o VMware Server Console e o VMware Server WebUI. O VMware Server Console permite um gerenciamento visual completo das máquinas virtuais, enquanto o WebUI permite apenas visualizar o status das máquinas virtuais (consumo de CPU e de memória, discos, etc) e executar comandos no VMware Server (reiniciar ou parar as máquinas virtuais, entre outros).

4 Ferramentas de *benchmarking*, testes e cenários

Neste trabalho, foram usadas algumas ferramentas e aplicações pra testar o desempenho das diferentes opções de virtualização de sistemas operacionais. Dada a ausência de recursos financeiros para a compra de *benchmarks* comerciais, as ferramentas escolhidas foram ferramentas *opensource*, baseadas em aplicações também *opensource* e sem restrição com relação à publicação dos resultados obtidos. Uma descrição de cada um dos *benchmarks*, bem como dos testes neles incluídos, vem a seguir na seção 4.1. Os cenários utilizados nos testes são descritos na seção 4.2.

4.1 Benchmarks e testes

4.1.1 Openbench

Segundo o desenvolvedor, Openbench [OpenBench, 2004] é uma tentativa de construção de um benchmark com as características do SPEC CPU2000 [SPEC, 2003]. O Openbench consiste em um conjunto de testes de compilação e de execução, baseados em pacotes de software já conhecidos no mundo *opensource*. Os softwares utilizados são: ***gzip*** e ***bzip2*** (ferramentas para compressão/descompressão de arquivos), ***OpenSSL*** e ***GnuPG*** (ferramentas para encriptação/decriptação de dados), ***libMAD*** (implementação livre do codec MP3) e ***Botan*** (biblioteca de segurança escita em C++). Esses softwares utilizados nos testes realizam tarefas que seriam executas normalmente por um usuário, em seu dia-a-dia. Os testes são os seguintes:

- gzip e bzip2: um arquivo tar (arquivo compactado) de tamanho 64MB e um arquivo binário de 16MB são compactados e descompactados em

vários níveis de compactação, do nível 1 (mais rápido, menor compactação) ao nível 9 (menos rápido, maior compactação) [Bzip2ManPage] [GzipManPage].

- OpenSSL : o utilitário "openssl speed" do pacote OpenSSL é usado para medir a performance. Esse utilitário gera *hashes* usando vários algoritmos de encriptação (md5,aes, des e outros) para vários blocos de dados de tamanhos diferentes.

- GnuPG: o comando "make check" é usado para testar a correta compilação do software e os arquivos usados nos testes gzip e bzip2 são encriptados.

- libMAD: três arquivos MP3 de diferentes tamanhos são decodificados para outro formato de áudio.

- Botan: o comando "make check" é utilizado para verificar a correta compilação do software.

4.1.2 SysBench

O *benchmark* SysBench é projetado para medir parâmetros importantes para um sistema executando um banco de dados sob carga intensiva [Sysbench, 2004]. Os testes medem os seguintes parâmetros: performance da CPU, performance de E/S de arquivos, performance do escalonador (*scheduler*), alocação de memória e velocidade de transferência da memória, performance de threads POSIX [1]e performance de um banco de dados.

- CPU: calcula números primos até um valor máximo especificado pelo usuário.

- Threads: vários threads são criados e eles competem entre si por

[1] POSIX é uma família de padrões abertos para sistemas operacionais.

mutexes[2].

- *Mutex*: vários threads são criados e eles também competem por *mutexes*. Entretanto, o lock adquirido sobre o *mutex* é ganho por um período muito curto de tempo. Sua finalidade é examinar a performance da implementação dos *mutexes.*

- Memória: a memória é lida ou escrita de forma seqüencial e os *threads* criados podem acessar tanto blocos de código globais quanto blocos de código locais.

- E/S de arquivos: vários tipos de operações de E/S em arquivos podem ser executadas. As operações incluem escritas, reescritas e leituras seqüenciais, e leituras e escritas randômicas de arquivos.

- Banco de dados: operações comuns de bancos de dados (criação de tabelas, remoção de tabelas, inserção de dados e outras) são executadas para medir a performance de uma base de dados.

4.1.3 netperf

O netperf é um benchmark usado para medir a performance de redes e dispositivos de redes [Netperf, 2007]. Os testes incluídos no netperf incluem testes de *troughput* e latência *end-to-end*. Esses testes realizam tranferências de dados via protocolos TCP e UDP, tanto transferências unidirecionais quanto do tipo *request/response*, seguindo o modelo básico cliente-servidor.

- Transferências de dados via protocolos TCP e UDP unidirecionais e do tipo *request/response* sobre os protocolos IPv4 e IPv6 usando a interface Sockets.

- Transferências de dados via protocolos TCP e UDP unidirecionais e do tipo *request/response* sobre os protocolos IPv4 e IPv6 usando a interface XTI.

- Transferências de dados a nível da camada de rede (*link-level*)

[2] Um mutex é um objeto utilizado em programação para permitir que vários *threads* compartilhem um determinado recurso ao mesmo tempo.

unidirecionais e do tipo *request/response* usando a interface DLPI.

- Sockets UNIX (*Unix Domain Sockets*).
- Transferências de dados via SCTP unidirecionais ou *request/response* sobre IPv4 e IPv6 usando a interface Sockets.

4.1.4 Unixbench

Unixbench [Unixbench, 1997] é um porte para UNIX do benchmark BYTEmark, criado pela equipe da revista BYTE.com [BYTE, 1994]. Os programas do unixbench incluem testes aritméticos, testes de sistema e uma implementação em C do programa Dhrystone [Weicker, R. , 1984].

- Testes aritméticos: *overhead* aritmético, aritmética de registradores, aritmética com números tipo *short, int, long, float* e *double*.
- Testes de sistema: *overhead* de chamadas de sistema (*syscall* e *execl*), *pipe throughput*, *pipe context sitch*, criação de processos (*spawn*), *throughput* do sistema de arquivos.
- Testes dhrystone com e sem registradores.
- Testes de compilação, de recursão e cálculos com o programa *dc* [GNU, 2006].

4.1.5 Ubench

O ubench [Ubench, 1998] consiste em duas partes. A primeira delas mede o desempenho da CPU através de uma série de operações matemáticas com números inteiros e números de ponto flutuante. A segunda parte realiza uma série de alocações e cópias de dados na memória. Tanto as operações matemáticas quanto as operações de memória são *senseless*, segundo os desenvolvedores do benchmark, no sentido de que as operações podem não tem nenhum sentido aparente.

- Testes aritméticos: operações aritméticas com números inteiros e números de ponto flutuante por três minutos, usando vários processos.

- Testes de memória: operações de alocação de dados na memória e de cópia de dados da memória por outros três minutos, usando também vários processos.

4.1.6 Httperf e autobench

O httperf [Mosberg, D. and Jin, T., 1998] é uma ferramenta usada para medir a performance de um servidor web, podendo gerar vários *workloads* no servidor HTTP durante sua execução. Autobench [Autobench, 2004] é um *wrapper* para o httperf, o que facilita o processo de benchmarking de um servidor web, inclusive gerando gráficos para posterior análise.

- httperf: faz várias requisições a um servidor http e calcula o número de requisições respondidas pelo servidor e a taxa de chegada das respostas.

- autobench: disparo de várias instâncias do httperf, aumentando o número de requisições por segundo a cada iteração.

4.1.7 SciMark

Scimark [Scimark, 2004] é um benchmark escrito em Java que realiza cálculos científicos e numéricos, gerando um *score* de Mflops (milhões de operações de ponto flutuante por segundo). Os testes realizam operações bastante realizadas em aplicações científicas e de engenharia, como transformadas rápidas de Fourier, multiplicações de matrizes esparsas e integração de Monte Carlo.

- Transformação rápida de Fourier: executa transformações de Fourier em um conjunto de 4K números complexos.

- Integração de Monte Carlo: aproxima o valor de Pi calculando a integral de y=sqrt(1-x^2) no intervalo [0,1].

- Multiplicação de matrizes esparsas: multiplicação de matrizes de tamanho 1000x1000, com 5000 números não-nulos.

- Fatorização LU de matrizes densas: cálculo da fatorização LU de matrizes com tamanho 100x100.

4.1.8 Bonnie++

O *benchmark* Bonnie++ [Bonnie, 2001] tem como objetivo realizar uma série de testes de disco e performance de sistemas de arquivos. Os testes compreendem leitura/escrita seqüencial e randômica.

- Testes de escrita seqüencial em arquivos: escrita de um caracter por vez (macro *putc()*), de um bloco de caracteres (função *write()*) e reescrita de blocos de caracteres (funções *read()*, *lseek()* e *write()*).

- Testes de leitura seqüencial de arquivos: leitura de um caracter por vez (macro *getc()*) e de blocos de caracteres (função *read()*).

- Testes de escrita/leitura randômicos: alguns processos são criados (em geral três) e executados em paralelos, lendo e escrevendo em partes randômicas do arquivo.

4.2 Cenários de testes

Os testes foram executados em configurações que imitam situações reais de uso de máquinas virtuais, além do cenário nativo, onde não temos nenhuma máquina virtual executando.

- Cenário 1 - "Nativo": testes são realizados no SO sem nenhuma máquina virtual em execução.
- Cenário 2: Duas máquinas virtuais executando em paralelo.

4.2.1 Cenário 1: Ambiente Nativo

No cenário 1, os testes são realizados em um ambiente não virtualizado. O sistema operacional utilizado foi o Debian GNU/Linux, versão 3.1. O sistema operacional utilizado executa serviços básicos de uma distribuição Linux (servidor X, servidor SSH, entre outros), além dos testes dos *benchmarks*. Para que tenhamos uma comparação justa entre esse cenário e os restantes, uma parte da memória total disponível (ver Cap 2) foi desabilitada. Este cenário apresenta a seguinte configuração:

- SO: Debian GNU/Linux 3.1
- Memória: 1GB
- Disco: duas partições
    - sda: 5120 MB - sistema de arquivos ext3
    - sdb: 512 MB - área de swap

4.2.2 Cenário 2: Duas Máquinas Virtuais em execução

No cenário 3, duas máquinas virtuais executam ao mesmo tempo. O sistema operacional instalado nas máquinas virtuais é o Debian GNU/Linux versão 3.1. Uma das máquinas virtuais executa os testes e serviços básicos de uma distribuição Linux (servidor X, servidor SSH, entre outros), enquanto a outra executa somente os serviços básicos. Nos testes realizados com o XenExpress, os SOs virtualizados foram instalados usando o template disponível na ferramenta de gerenciamento. As duas máquinas virtuais apresentam a mesma configuração de hardware, como discos e quantidade de memória. O cenário 3 apresenta três configurações:

Configuração 1:

- VMM: XenExpress 3.1
- SO *Host*: Não aplicável (O VMM é o próprio SO)
- SOs *Guests*: Debian GNU/Linux 3.1, kernel 2.6.16.29, modificado para suportar o VMM Xen
- Memória SO *host*: 512 MB
- Memória SOs *guests*: 1GB cada um
- Discos Virtuais: duas partições
    - sda: 5120 MB - sistema de arquivos ext3
    - sdb: 512 MB - área de swap

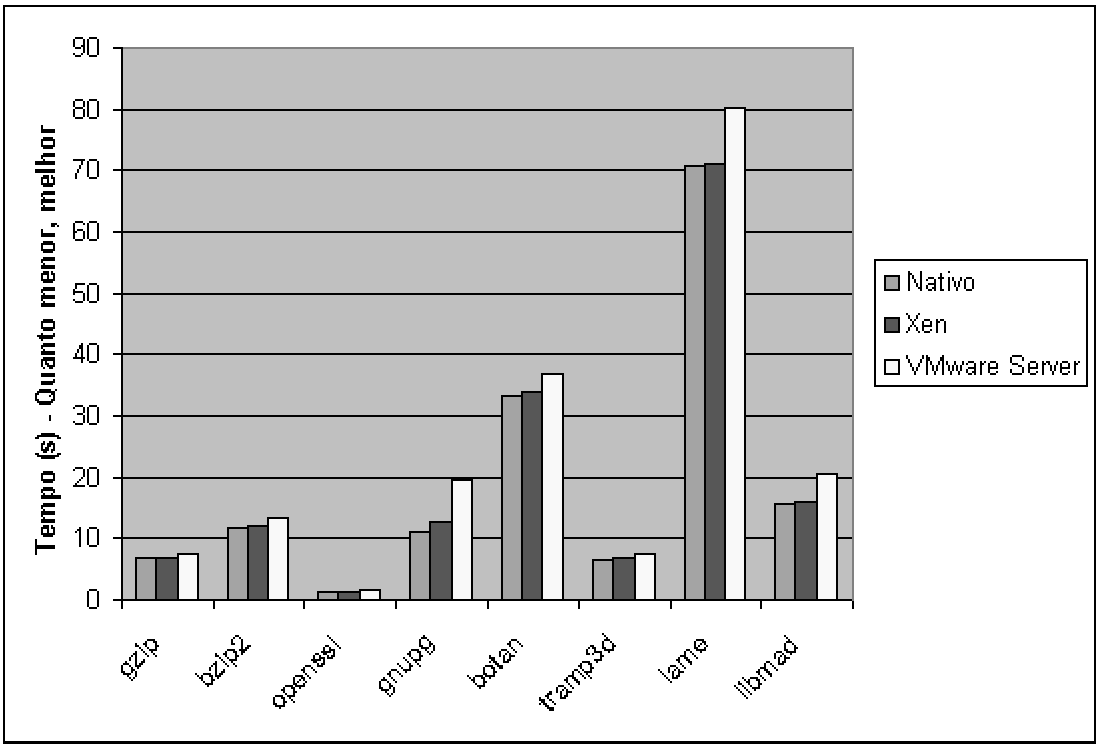
Configuração 2:

- VMM: VMware Server (Free) 1.0.2
- SO host: Debian GNU Linux 3.1
- SOs guests: Debian GNU/Linux 3.1
- Memória SO *host*: 512 MB
- Memória SOs *guests*: 1 GB cada um
- Discos Virtuais: duas partições
    - sda: 5120 MB - sistema de arquivos ext3
    - sdb: 512 MB - área de swap

5 Resultados

5.1 Openbench

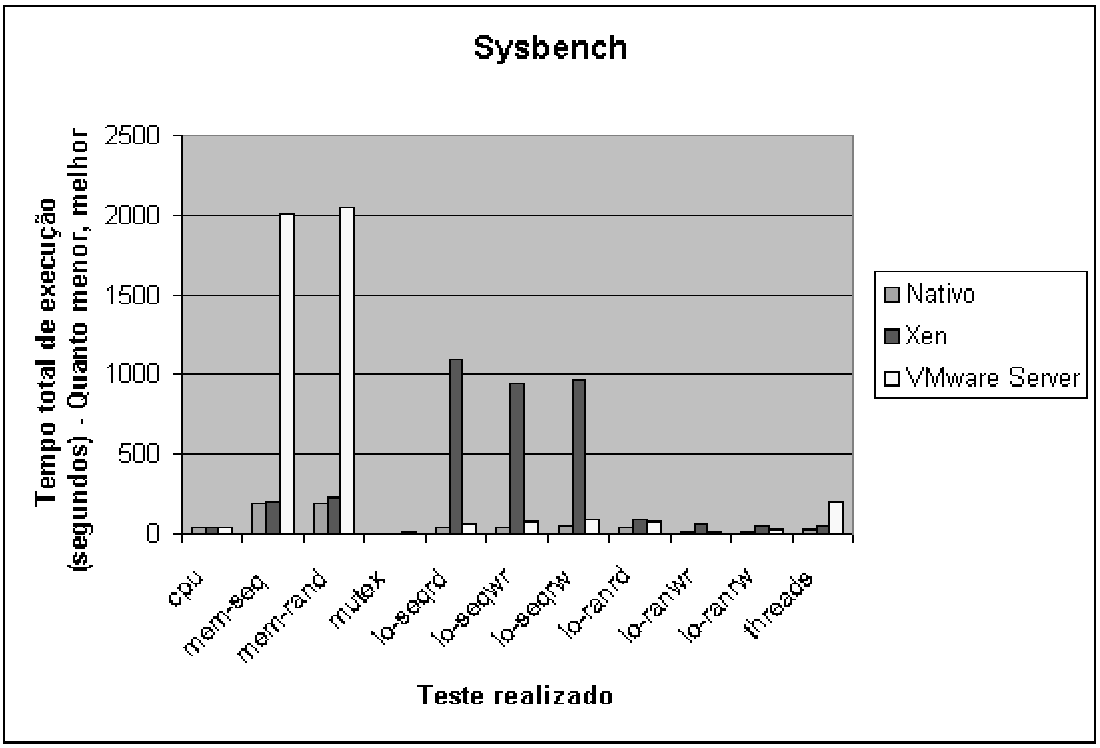
Os testes realizados com o Openbench mostram que, para as atividades realizadas normalmente por um usuário, que não exigem um uso intensivo de entrada e saída de dados (um servidor de arquivos, por exemplo), a perda de performance não é muito significativa. Os resultados gerais mostram que o Xen[3] teve uma performance superior à do VMware, aproximando-se bastante da performance nativa.



[3] Deste ponto em diante, os produtos serão chamados pelos nomes: Xen para o XenExpress e VMware para o VMware Server.

5.2 Sysbench

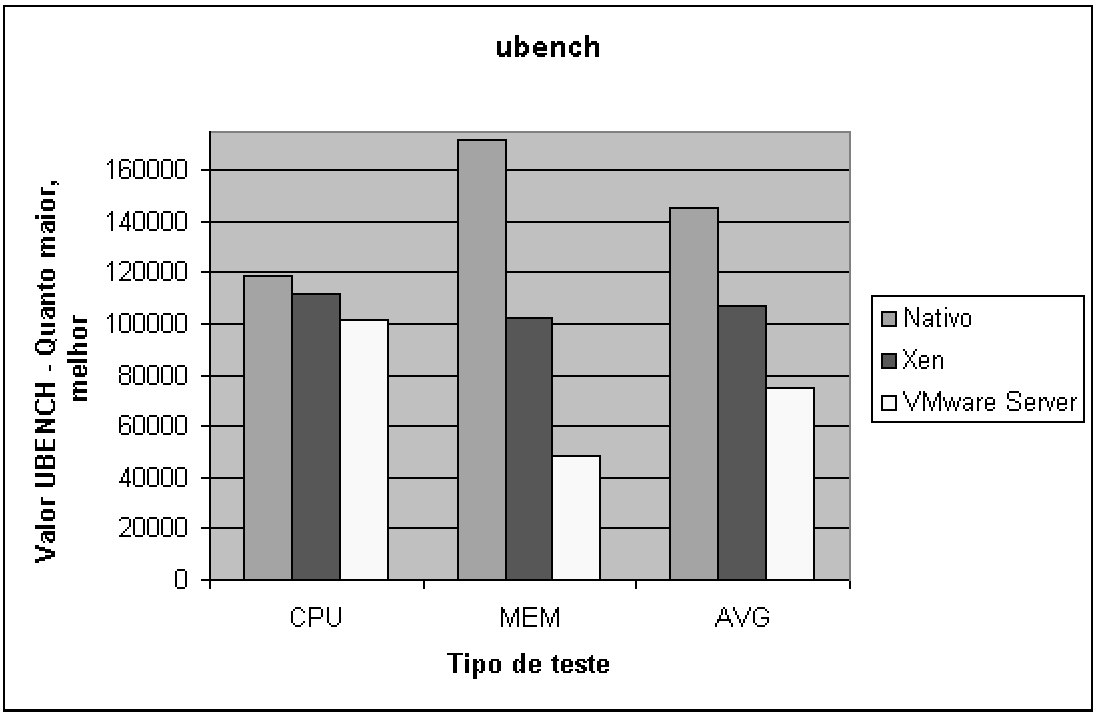
Para o *benchmark* Sysbench, os testes apresentaram resultados interessantes. No item memória, o VMware apresentou um desempenho muito inferior ao do Xen. Já em relação aos testes de leitura e escrita em disco, o Xen teve uma performance bastante inferior.



Durante a realização dos testes no Xen, foi notado um fato interessante. Quando os testes de entrada/saída estavam sendo executados, o sistema parava de responder aos comandos por algum tempo, só retornando o controle ao usuário algum tempo depois. Esse fato não foi observado no VMware.

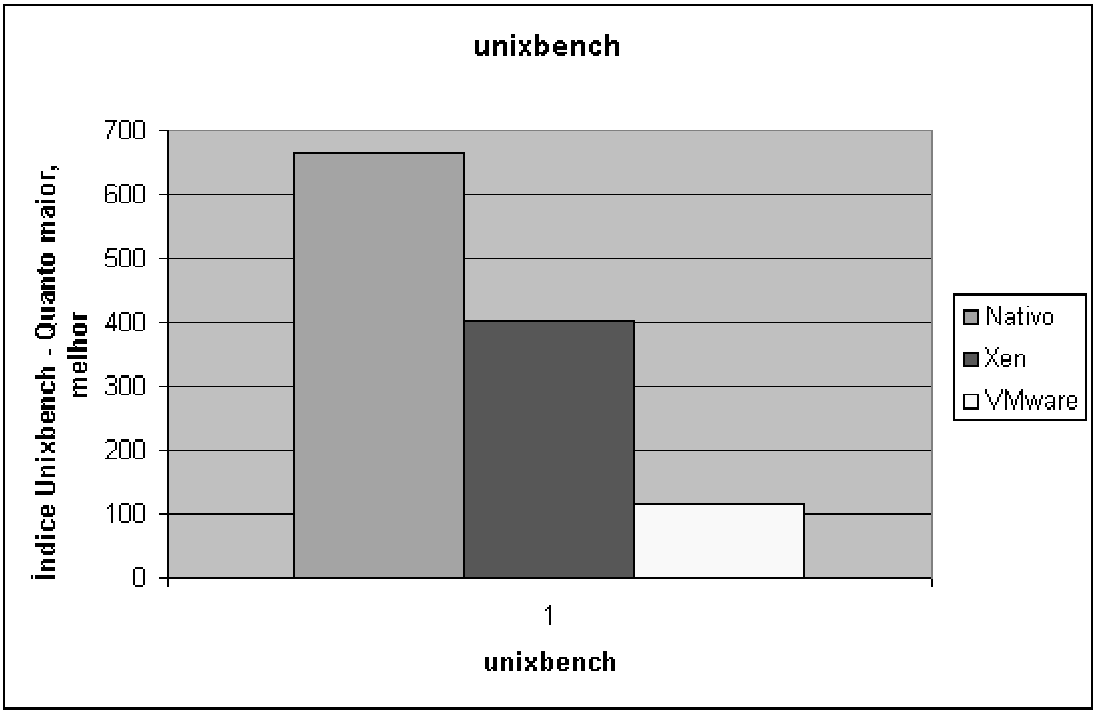
5.3 Ubench

Nos testes do Ubench, o VMware novamente apresentou uma performance inferior à do Xen. A diferença foi percebida significativamente no teste de memória, uma vez que no teste de CPU a performance do Xen não foi muito superior à do VMware.



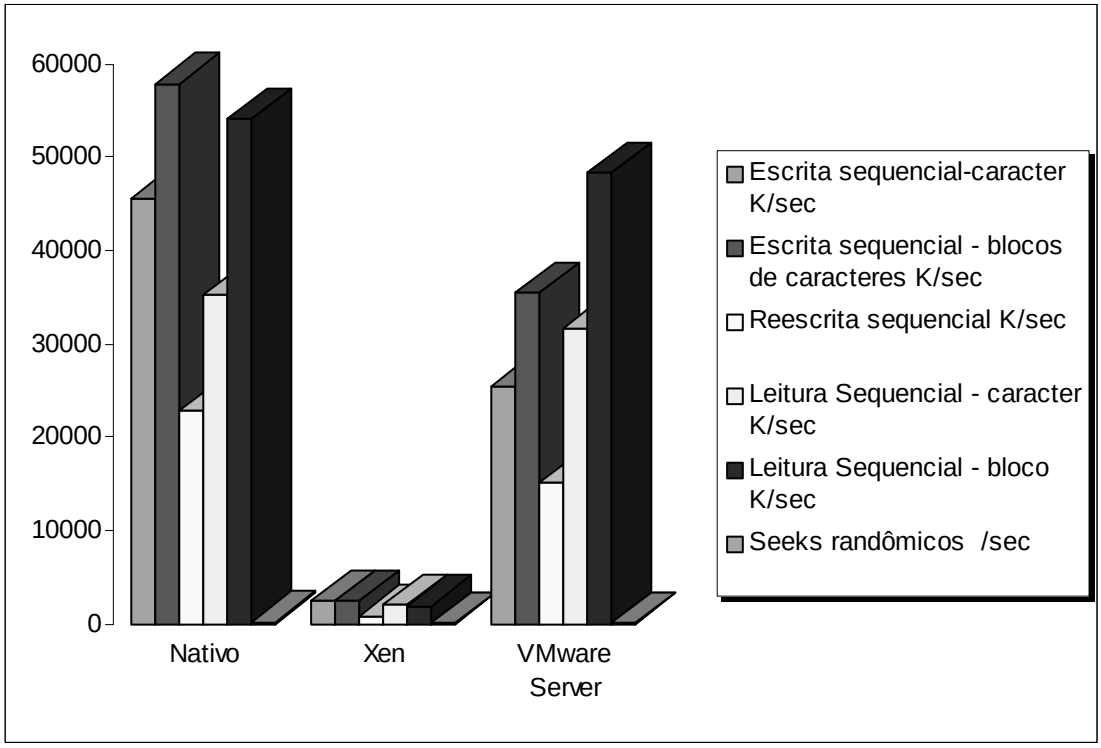
5.4 UnixBench

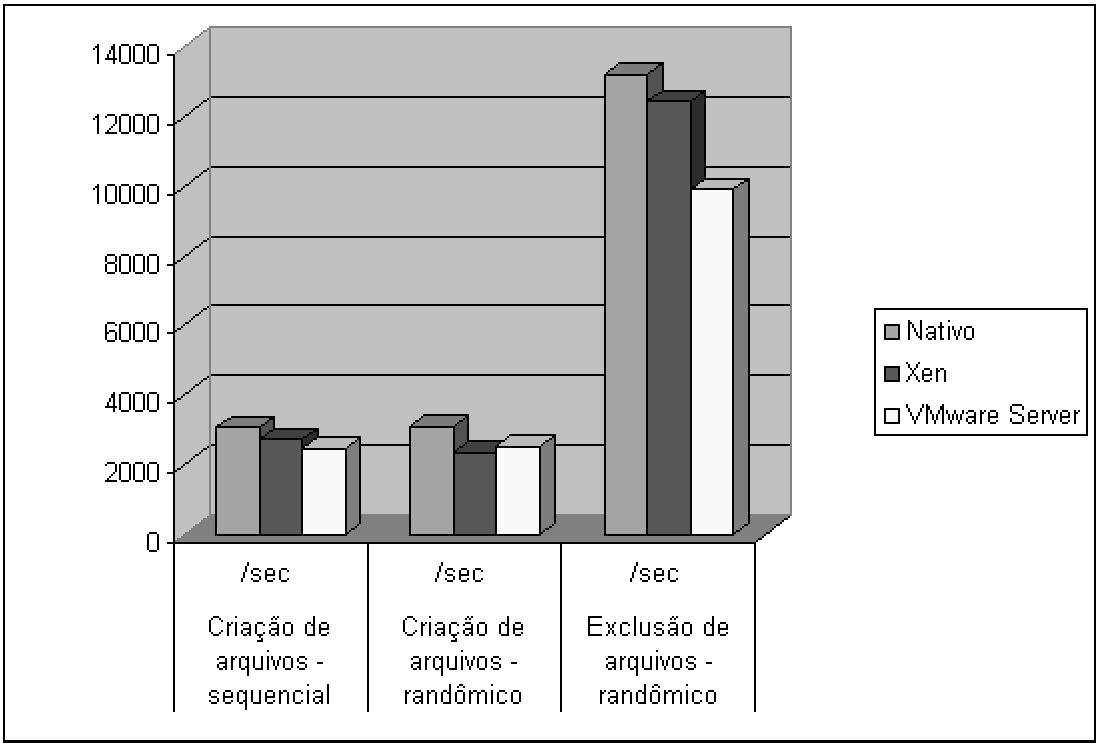
Nos resultados do UnixBench, novamente o VMware apresentou resultados inferiores aos do VMware, mesmo apresentando em alguns dos testes do *benchmark* resultados superiores aos do Xen (testes de leitura/escrita de arquivos).



5.5 Bonnie++

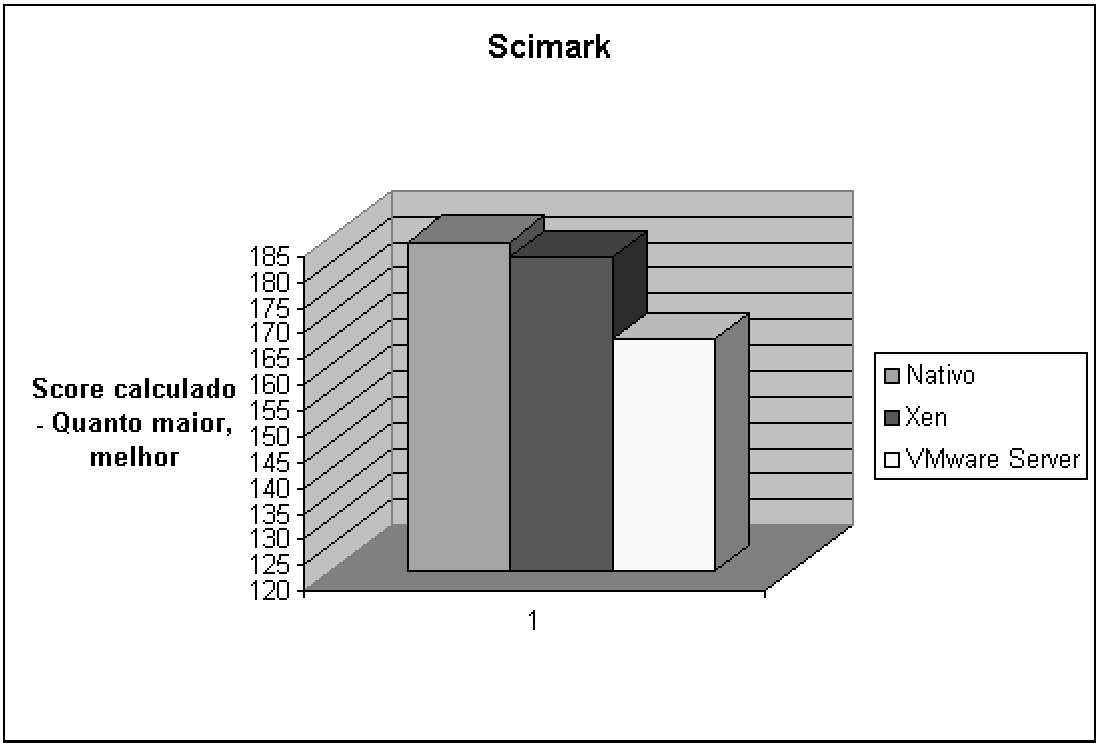
Os testes do Bonnie++ confirmaram o que havia sido constatado pelos testes anteriores. O Xen apresenta um comportamento de certa forma pífio em relação ao VMware, quando em operações de escrita/leitura de arquivos. Já nas operações simples de criação/remoção de arquivos, o VMware tem um desempenho levemente inferior.




14000
12000
10000
8000
6000
4000
2000
0
/sec
/sec
/sec
Criação de arquivos - sequencial
Criação de arquivos - randômico
Exclusão de arquivos - randômico
Nativo
Xen
VMware Server

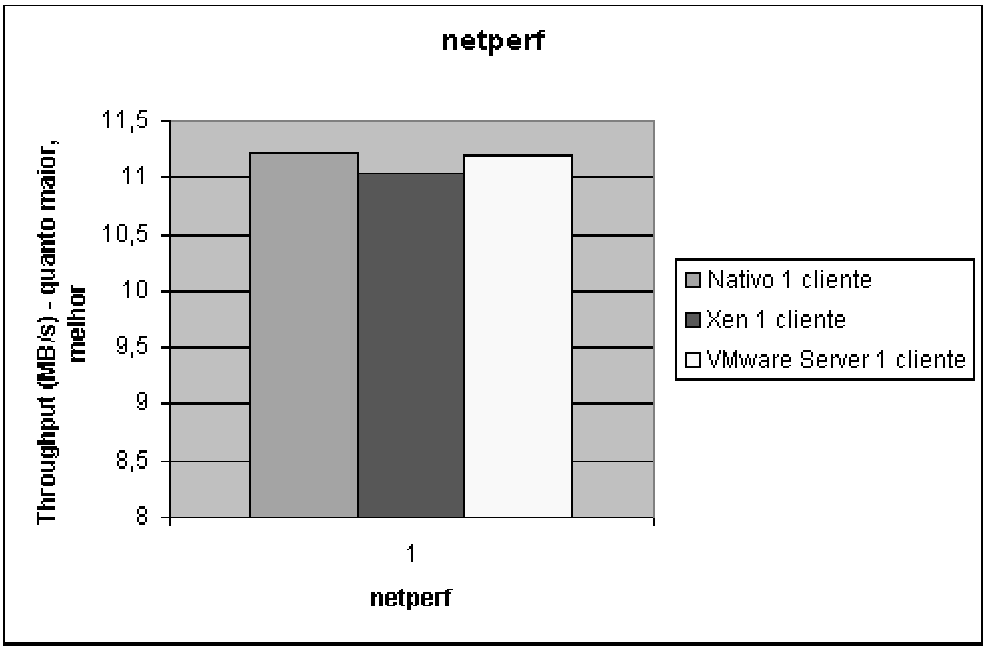
5.6 Scimark

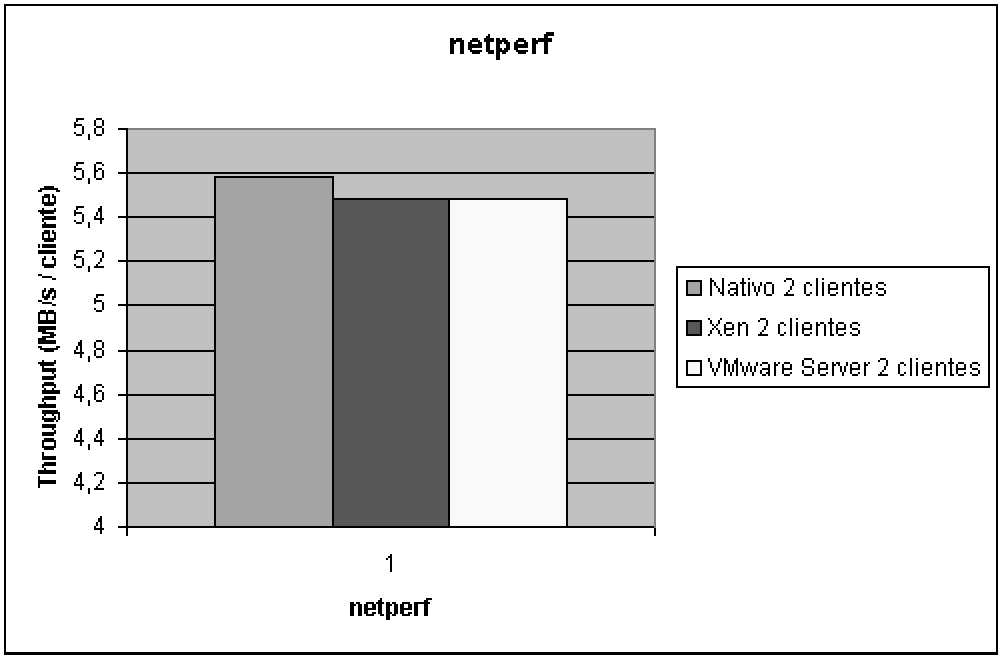
Como na maioria dos resultados anteriores, o Xen apresentou resultados superiores aos do VMware. Os testes do Scimark fazem bastante uso de memória e, como foi notado nos testes anteriores, o VMware não se comporta muito bem com o uso extensivo da memória.



5.7 Netperf

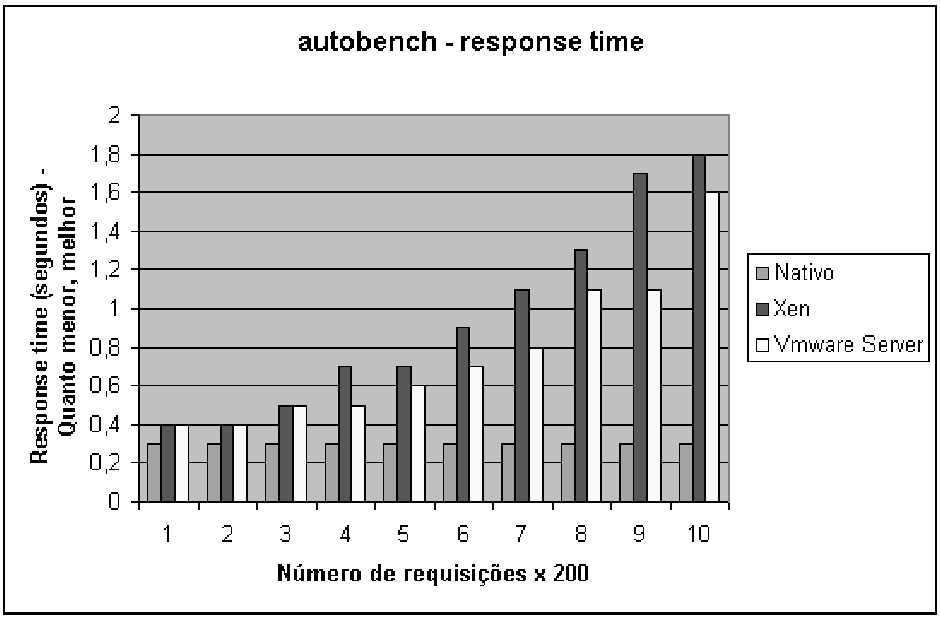
O netperf apresentou também resultados interessantes. Tanto o VMware quanto o Xen tiveram desempenho semelhante, com uma pequena vantagem do VMware. Essa vantagem pode ser oriunda do fato que o VMware instala drivers de rede no sistema *host*, e as máquinas virtuais utilizam o hardware diretamente. Como também era de se esperar, o fluxo de dados quando temos duas instâncias de clientes fazendo requisições ao servidor, o fluxo de dados é metade do fluxo de um cliente apenas.

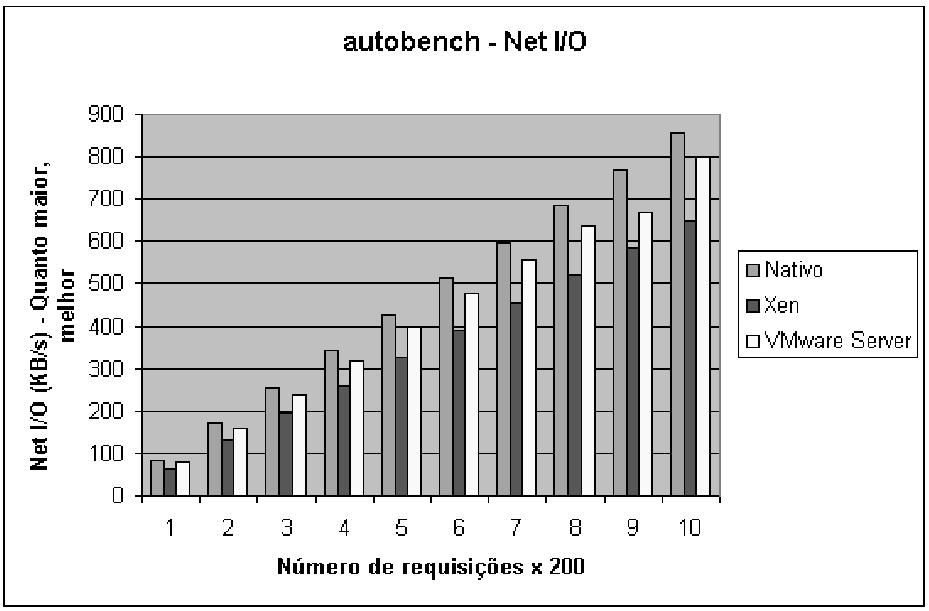



netperf
Throughput (MB/s / cliente)
5,8
5,6
5,4
5,2
5
4,8
4,6
4,4
4,2
4
1
netperf
Nativo 2 clientes
Xen 2 clientes
VMware Server 2 clientes

5.8 Autobench

Os testes realizados com a ferramenta autobench mostram certa vantagem do VMware em relação ao Xen. A presença de drivers instalados no SO *host*, como indicado na seção anterior fornece indícios para perceber que a performance do VMware pode ser oriunda da utilização desses drivers. Ainda assim, esses resultados mostram que essa abordagem do VMware, de carregar um driver no SO *host* é bastante válida.





6 Conclusões

Neste trabalho, vimos os principais conceitos relacionados a virtualização. Além disso, vimos alguns dos *benchmarks* utilizados na comunidade opensource e os utilizamos para comparar alguns dos principais produtos de virtualização. Nos cenários testados, vimos que há uma paridade entre as duas opções em relação ao número de testes em que um deles se sobressaiu sobre o outro. Assim, não é possível indicar um vencedor, a menos que se saiba quais as principais tarefas a ser realizadas pelas máquinas virtuais. Em tarefas de intensivo I/O e de uso da rede, o Xen não apresenta resultados muito satisfatótios em relação ao VMware. Já o VMware não apresenta resultados satisfatórios nos testes de uso intensivo de memória.

Como contribuição principal, este trabalho mostrou as diferenças entre as implementações dos principais concorrentes, podendo servir como um estudo inicial para o projeto de implementação de um ambiente de múltiplas máquinas virtuais no Centro de Informática da UFPE.

6.1 Dificuldades encontradas

Durante o decorrer do desenvolvimento deste trabalho, algumas dificuldades foram encontradas. Algumas delas têm a ver com aspectos financeiros, como a aquisição de licenças para os produtos mais promovidos pelos fabricantes de VMMs e ausência de *benchmarks opensource* com o reconhecimento (fora da comunidade *opensource*) dos benchmarks da suíte SPEC. Devido a esses e outros fatores, a comparação entre os produtos ficou restrita a somente dois deles. Para uma comparação mais completa, seria necessário o uso mais produtos, incluindo os produtos “top-de-linha” das empresas fabricantes dos VMMs. Outro fator que dificultou o andamento deste trabalho foi a indisponibilidade inicial de hardware para a realização dos testes. Depois de conseguido esse hardware, foi verificado que o

mesmo não é compatível para a utilização do software VMware ESX Server, principal produto “enterprise” do fabricante VMware. Esse fato fez com que o número de configurações dos cenários utilizados ficasse inferior ao planejado inicialmente.

6.2 Trabalhos Futuros

Como trabalhos futuros, pode ser realizado um estudo mais aprofundado sobre os motivos de os resultados encontrados com o Xen nos testes de rede e de I/O, com uma proposta de solução desses problemas.

Anexos

Tabelas utilizadas para geração dos gráficos

Openbench

	gzip	bzip2	openssl	gnupg	botan	tramp3d	lame	libmad
Nativo	6,79	11,775	1,26156	11,1	33,15585	6,38	70,66	15,72
Xen	6,885	11,92	1,39326	12,59	33,92804	6,715	71,04	16,125
VMware Server	7,45	13,51	1,537	19,535	37	7,51	80,25	20,395

Sysbench

	Nativo	Xen	VMware Server
cpu	38,6846	39,8429	43,854
mem-seq	193,5548	202,2979	2006,182
mem-rand	187,9617	219,9	2046,845
mutex	0,33658	0,5401	7,4152
io-seqrd	41,9631	1088,536	65,2203
io-seqwr	39,1037	943	78,4631
io-seqrw	49,7306	962	83,0728
io-ranrd	42,8474	90,2083	72,584
io-ranwr	7,4674	62,4833	9,3257
io-ranrw	18,6374	45,9803	21,247
threads	22,7328	52,0703	201,1732

Ubench

	CPU	MEM	AVG
Nativo	119074	171583	145328,5
Xen	111629	102615	107122
Vmware Server	101555	48514	75034

Unix bench

Nativo	664,9
Xen	404,2
VMware	115,1

Bonnie

	Escrita sequencial- caracter K/sec	Escrita sequencial - blocos de caracteres K/sec	Reescrita sequencial K/sec	Leitura Sequencial - caracter K/sec	Leitura Sequencial - bloco K/sec	Seeks randômicos /sec
Nativo	45605	57854	22861	35234	54086	204,8
Xen	2486	2482	894	2078	1971	137,4

VMware Server	25477	35503	15150	31754	48339	190,2

Scimark

Nativo	183,91
Xen	181,06
VMware Server	165,23

Netperf

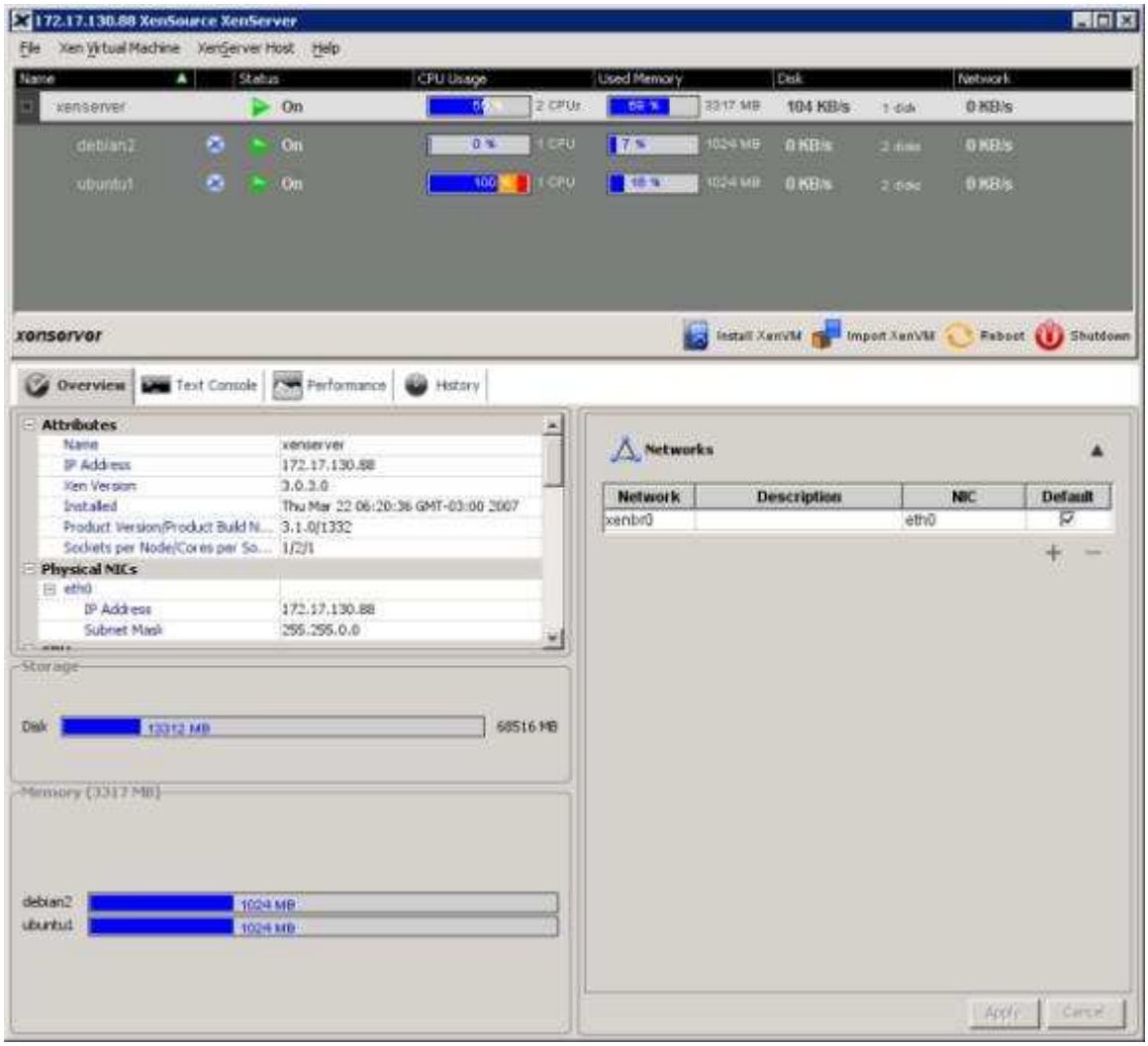
Nativo	1 cliente	11,21
Xen	1 cliente	11,04
VMware Server	1 cliente	11,2
Nativo	2 clientes	5,58
Xen	2 clientes	5,48
VMware Server	2 clientes	5,485

Autobench

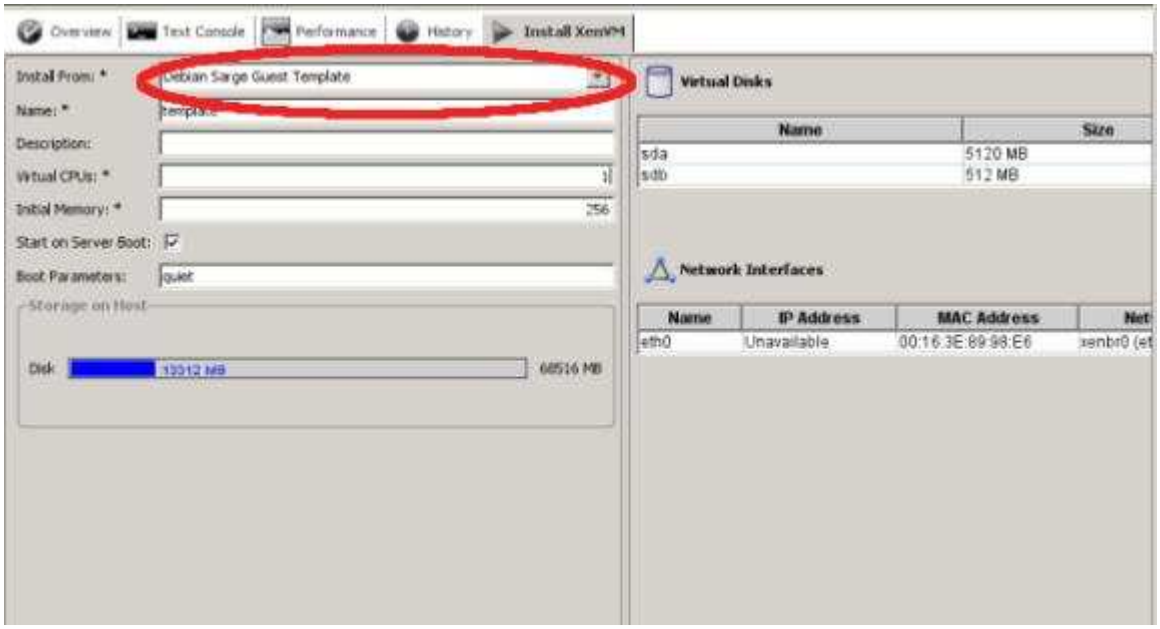
Nativo										
Número de requisições	200	400	600	800	1000	1200	1400	1600	1800	2000
Response Time	0,3	0,3	0,3	0,3	0,3	0,3	0,3	0,3	0,3	0,3
Net I/O	85,6	171,1	256,7	342,2	427,8	513,3	598,9	684,5	770	855,5
Xen										
Número de requisições	200	400	600	800	1000	1200	1400	1600	1800	2000
Response Time	0,4	0,4	0,5	0,7	0,7	0,9	1,1	1,3	1,7	1,8
Net I/O	65,1	130,1	195,2	260,2	325,3	390,3	455,4	520,4	585,5	650,4
VMware Server										
Número de requisições	200	400	600	800	1000	1200	1400	1600	1800	2000
Response Time	0,4	0,4	0,5	0,5	0,6	0,7	0,8	1,1	1,1	1,6
Net I/O	79,9	159,8	239,7	318,7	399,2	479,3	559,2	639,1	668,9	798,8

Emulador SNES9x

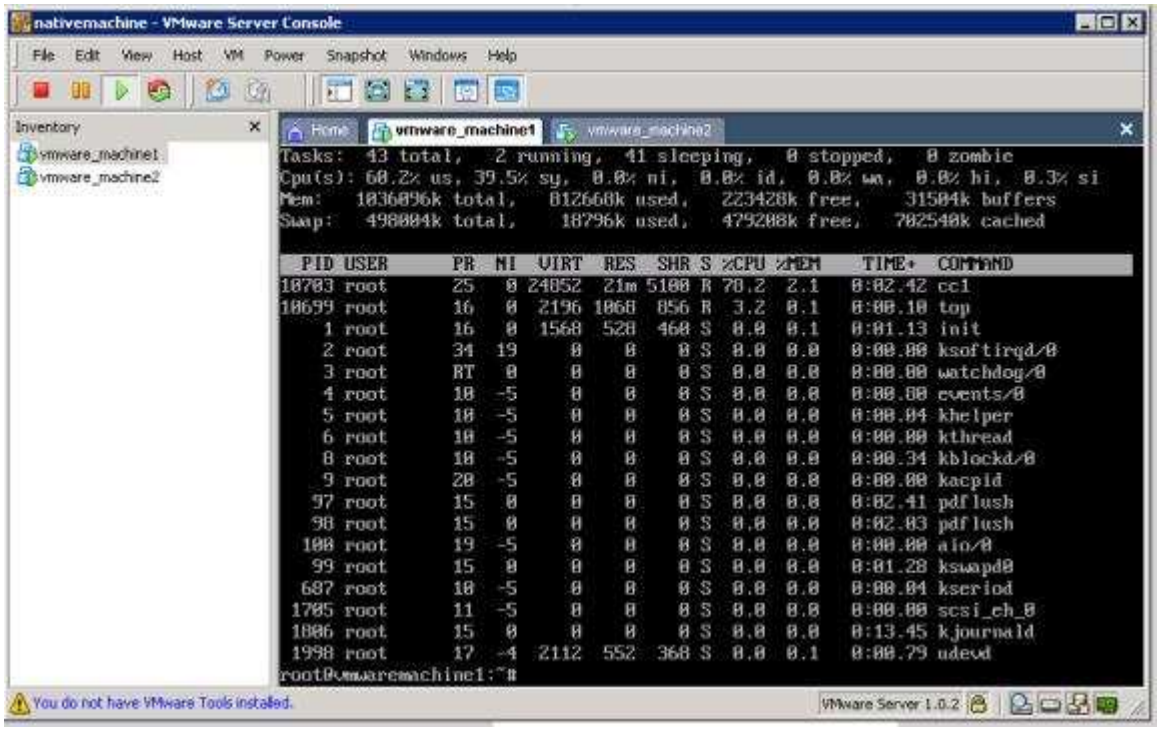
Interface de gerenciamento do XenExpress



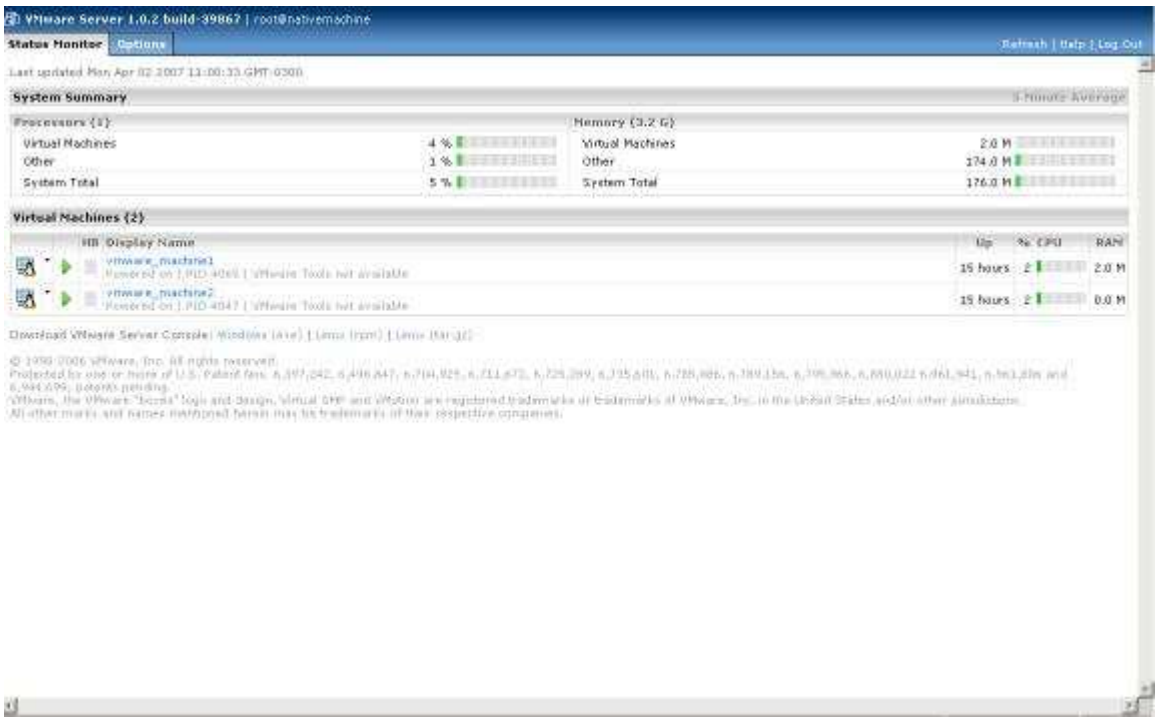
Instalação de SO *guest*, com template para debian



Interface de gerenciamento do VMware Server , VMware Server Console,



VMware Server WebUI



Referências

[Bochs, 2007] *bochs: The Open Source IA-32 Emulation Project.* 2007. Disponível em http://bochs.sourceforge.net, acesso em 22/03/2007.

[Eweek, 2007a] *Adoption of Virtualization Continues to Grow: Report.* 2007. Disponível em: http://www.eweek.com/article2/0%2C1759%2C2094148%2C00.asp?kc=EWRSS03119TX1K0000594, acesso em 26/02/2007

[Wikipedia, 2007] *Virtualization – Wikipedia*. 2007. Disponível em: http://en.wikipedia.org/wiki/Virtualization , acesso em 26/02/2007

[Bellard, 2005] Bellard, F. *QEMU, a fast and portable Dynamic Translator*. 2007. Disponível em: http://www.usenix.org/events/usenix05/tech/freenix/full_papers/bellard/bellard_html, acesso em 26/02/2007.

[ePSXe, 2003] *ePSXe Homepage*. 2003. Disponível em http://www.epsxe.com, acesso em 13/03/2007.

[Channel, 2005] *Channel Web Network - Linux 2.6 Kernel To include Xen Virtualization Technology*. 2005. Disponível em http://www.crn.com/software/59300297, acesso em 11/03/2007

[VMware, 2007] ESX vs XEN Hypervisor Performance. 2007. Disponível em: http://www.vmware.com/pdf/hypervisor_performance.pdf, acesso em 11/03/2007

[XenSource, 2007] *XenEnterprise vs. ESX Benchmark Results: Performance Comparison of Commercial Hypervisors*. 2007. Disponível em: http://blogs.xensource.com/simon/wp-content/uploads/2007/03/hypervisor_performance_comparison_1_0_3_no_esx-data.pdf, acesso em 11/03/2007

[Bjerke, 2005] Bjerke, H. K. F. *HPC Virtualization with Xen on Itanium*, 2005. Disponível em: http://openlab-mu-internal.web.cern.ch/openlab-mu-internal/Documents/2_Technical_Documents/Master_thesis/Thesis_HarvardBjerke.pdf, acesso em 11/03/2007.

[Laureano, 2006] Laureano, M. *Máquinas Virtuais e emuladores. Conceitos,Técnicas e Aplicações*. 2006. Novatec Editora, São Paulo.

[Creasy, 1981] *Creasy, R. J. "The origin of the VM/370 time-sharing system", IBM Journal of Research & Development*. 1981. Disponível em: http://www.research.ibm.com/journal/rd/255/ibmrd2505M.pdf, acesso em 15/03/2007.

[Goldberg, 1973] R.P. Goldberg. *Architectural Principles for Virtual Computer Systems*. 1973. Ph.D. thesis, Harvard University, Cambridge, Mass., ESD-TR-73-105, HQ Electronics Systems Division, Hanscom Field, Bedford, Massachusetts.

[King, 2003] Samuel T. King, George W. Dunlap, Peter M. Chen. *Operating System Support for Virtual Machines.* 2003. Proceedings of the 2003 USENIX Technical Conference.

[Barham, 2003] Barham, P. et al. *Xen and the Art of Virtualization.* 2003. Disponível em: http://www.cl.cam.ac.uk/research/srg/netos/papers/2003-xensosp.pdf, acesso em 15/03/2007.

[Microsoft, 2007] *Microsoft Virtual PC 2007*. Disponível em http://www.microsoft.com/windows/products/winfamily/virtualpc/default.mspx. Acesso em 13/03/2007.

[Microsoft, 2006] *Microsoft Press - Microsoft and XenSource to Develop Interoperability for Windows Server "Longhorn" Virtualization.* Disponível em: http://www.microsoft.com/presspass/press/2006/jul06/07-17MSXenSourcePR.mspx, acesso em 13/03/07.

[Tijms, 2000] Tijms, A. *Binary translation, classification of emulators*. Technical Report. Leiden Institute for Advanced Computer Science - University Leiden, 2000.

[Jones, 2007] Jones, M. T. *Virtual Linux - IBM Developer's Works*. 2007. Disponível em http://www-128.ibm.com/developerworks/linux/library/l-linuxvirt/?ca=dgr-lnxw01 Virtual-Linux, acesso em 15/03/07.

[Leung, 2006] Leung, F.; Neiger, G.; Rodgers, D.; Santoni, A.; Uhlig, R. *"Intel Virtualization Technology: Hardware Support for Efficient Processor Virtualization." Intel Technology Journal*. 2006. Disponivel em: http://www.intel.com/technology/itj/2006/v10i3/, acesso em 17/03/2007.

[Nanda and Chiueh, 2005] Nanda, S. and Chiueh, T. *A Survey on Virtualization Technologies.* 2005. Technical Report. Department of Computer Science. University at Stony Brook, NY.

[King, 2002] King, Samuel T. and Chen, Peter M. *Operating System Extensions to Support Host Based Virtual Machines*. 2002. Technical Report CSE-TR-465-02. University of Michigan.

[Weicker, R. 1984] Weicker, Reinhold P. *Dhrystone: a synthetic systems programming benchmark*. 1984. Communications of the ACM. Volume 27 , Issue 10.

[Mosberger, D. and Jin, T., 1997] Mosberger,D. and Jin, T. httperf: A Tool for Measuring Web Server Performance**.** Disponível em http://www.hpl.hp.com/personal/linux/httperf/wisp98/httperf.pdf, acesso em 13/03/2007.

[SNES9x, 1999] SNES9x.com. Homepage do emulador SNES9x. Disponível em http://www.snes9x.com , acesso em 13/03/2007.

[ZSNES, 2001] ZSNES Home Page. Disponível em http://www.zsnes.com , acesso em 13/03/2007.

[Wine, 1993] Wine HQ. Disponível em http://www.winehq.com , acesso em 13/03/2007.

[Chernoff and Hookway, 1997] Chernoff,A. and Hookway, R. *DIGITAL FX!32 Running 32-Bit x86 Applications on Alpha NT*. 1997. USENIX Windows NT Workshop. Disponível em: http://www.usenix.org/publications/library/proceedings/usenix-nt97/full_papers/chernoff/chernoff.pdf, acesso em 02/04/2007.

[Sun, 1999] The Java HotSpot Performance Engine: An In Depth Look. Disponível em http://java.sun.com/developer/technicalArticles/Networking/HotSpot, acesso em 02/04/2007.

[Intel, 2007] Intel Virtualization Technology. Disponível em http://www.intel.com/cd/ids/developer/asmo-na/eng/dc/enterprise/technologies/221962.htm , acesso em 02/04/2007.

[AMD, 2007] Introducing AMD Virtualization. Disponível em: http://www.amd.com/us-en/Processors/ProductInformation/0,,30_118_8796_14287,00.html, acesso em 02/04/2007.

[OpenBench, 2004] ExactCODE: OpenBench. Disponível em http://www.exactcode.de/site/open_source/openbench , acesso em 02/04/2007.

[SPEC, 2003] SPEC CPU 2000. Disponível em http://www.spec.org/cpu2000, acesso em 02/04/2007.

[Netperf, 2007] Netperf Home Page. Disponível em http://www.netperf.org/netperf/NetperfNew.html , acesso em 13/03/2007.

[Unixbench, 1998] Unixbench. Disponível em http://www.tux.org/pub/tux/niemi/unixbench, acesso em 13/03/2007.

[BYTE, 2007] BYTE.com. DIsponível em http://www.byte.com , acesso em 13/03/2007.

[GNU, 2006] GNU bc. Disponível em http://www.gnu.org/software/bc , acesso em 02/04/2007.

[Ubench, 1998] Ubench. Disponível em http://www.phystech.com/download/ubench.html , acesso em 13/03/2007.

[Autobench, 2004] Autobench. Disponível em http://www.xenoclast.org/autobench , acesso em 17/03/2007.

[Scimark, 2004] Java Scimark2. Disponível em http://math.nist.gov/scimark2, acesso em 22/03/2007.

[Bonnie, 2001] Bonnie++. Disponível em http://www.coker.com.au/bonnie++, acesso em 22/03/2007.