LNCC - LABORATÓRIO NACIONAL DE COMPUTAÇÃO CIENTÍFICA
ISTCC - INSTITUTO SUPERIOR DE TECNOLOGIA EM CIÊNCIAS DA COMPUTAÇÃO

Virtualização de Sistemas Operacionais

Por

Rodrigo Ferreira da Silva

Orientador

Prof. Fábio Borges de Oliveira

Petrópolis - RJ, Brasil

Dezembro de 2007

LNCC - LABORATÓRIO NACIONAL DE COMPUTAÇÃO CIENTÍFICA
ISTCC - INSTITUTO SUPERIOR DE TECNOLOGIA EM CIÊNCIAS DA COMPUTAÇÃO

Virtualização de Sistemas Operacionais

Monografia apresentada ao curso de Graduação em Tecnologia da Informação e da Comunicação do ISTCC – Instituto Superior de Tecnologia em Ciências da Computação de Petrópolis, como pré-requisito para obtenção do título de graduado em Tecnologia da Informação e da Comunicação.

FOLHA DE APROVAÇÃO

Virtualização de sistemas operacionais

Por

Rodrigo Ferreira da Silva

Monografia apresentada à Comissão Examinadora como requisito da Disciplina **Projeto Final**:

Fábio Borges de Oliveira

Laboratório Nacional de Computação Científica - LNCC

Instituto Superior de Tecnologia em Ciências da Computação – ISTCC

Eduardo Lúcio Mendes Garcia

Laboratório Nacional de Computação Científica - LNCC

Rogério Albuquerque de Almeida

Laboratório Nacional de Computação Científica - LNCC

Wagner Vieira Léo

Laboratório Nacional de Computação Científica - LNCC

Petrópolis - RJ, Brasil

Dezembro de 2007

AGRADECIMENTOS

A Deus acima de tudo, à minha noiva Juliana que sempre me deu apoio nas horas difíceis, aos meus pais que sempre acreditaram e investiram em meus estudos, ao meu orientador Fábio pelo tempo disponibilizado entre as suas atividades, a Tânia e ao pessoal da secretaria do IST sempre dando uma força extra aos alunos, ao Lincoln, ao professor Sérgio da Tiny English Course de Petrópolis, aos colegas Bruno, Alex, Luciana e Mariana, aos professores Luis Fernando, Augusto e Otávio (*in memoriam*), e a todos que me ajudaram nesta difícil caminhada.

RESUMO

A virtualização de sistemas operacionais é uma tecnologia que vem ganhando espaço nos últimos anos e cuja principal proposta é particionar os recursos do *hardware* de forma que ele execute vários sistemas operacionais (iguais ou diferentes) e suas aplicações de forma simultânea e totalmente isoladas entre si. Com a virtualização podemos fazer um melhor aproveitamento dos recursos computacionais novos ou existentes, reduzindo a freqüente ociosidade desses recursos em momentos do dia e do mês. Este trabalho tem como objetivo, apresentar os principais conceitos, bem como algumas das características técnicas, além de exemplificar outras importantes utilizações desta tecnologia. Ele descreve também, as arquiteturas dos principais *softwares* de virtualização disponíveis na atualidade, inclusive com detalhes de implementação.

Palavras-Chave: Virtualização, Máquinas Virtuais, Sistemas Operacionais, Emuladores.

ABSTRACT

The operating systems virtualization is a technology that, is getting large in the last years and which main proposal is divide the resources of the hardware to make it execute several operating systems (different or the same), their applications in a simultaneous way and totally isolated among themselves. With the virtualization we can make a better use of the computer resources: new or existent, reducing the frequent idleness of those resources in moments of the day and of the month. This work has the intention, to present the main concepts, as well as some of the technical characteristics, besides exemplifying other important uses of this technology. It also describes, the architectures of the main softwares of virtualization available at the present time, including implementation details.

Keywords: Virtualization, Virtual Machines, Operating systems, Emulators.

SUMÁRIO

INTRODUÇÃO

Os sistemas computacionais tradicionais vêm se baseando há alguns anos no modelo *Hardware* - Sistema Operacional - Aplicações. Todavia, nesse modelo há um problema: uma aplicação geralmente só executa sobre o sistema operacional para qual ela foi escrita. Assim somos obrigados a ter um sistema operacional por vez executando em determinado *hardware*, e somente as aplicações que executam sobre esse sistema poderão ser executadas nesse *hardware*.

A virtualização de sistemas operacionais é um meio para reduzir a importância do sistema operacional. Ela visa permitir que um *hardware* possa executar vários sistemas operacionais iguais ou distintos, de uma forma simultânea e isolados entre si. Para isto utiliza-se de técnicas avançadas de abstração e emulação, sempre mantendo esforços para prover o máximo de segurança, desempenho e confiabilidade dos meios envolvidos.

Além disso, é também uma proposta para consolidação de servidores, redução do consumo de energia elétrica, produção de calor, espaço físico e manutenção de equipamentos. Benefícios estes que garantem um melhor aproveitamento dos recursos computacionais existentes, assim, gerando menores custos e menor degradação ambiental.

O presente trabalho consiste em apresentar os principais pontos desta tecnologia que vem ganhando bastante espaço nos últimos anos. Ele visa principalmente servir de base a pessoas que estão ingressando no estudo do tema, e podendo ser utilizado como ponto de partida a pesquisas mais específicas na área no futuro.

Este trabalho está dividido da seguinte forma: principais conceitos da tecnologia, principais utilizações da tecnologia, a arquitetura e as diferenças das técnicas de virtualização existentes, e as características dos principais *softwares* de virtualização atuais. É importante salientar que alguns desses *softwares* foram testados de forma

prática, e ao final de cada apresentação do *software* neste trabalho, há um relato sobre como foi feito o teste e uma rápida conclusão.

Na última seção há uma conclusão geral e sugestões para trabalhos futuros.

CAPÍTULO I

CONCEITO DE VIRTUALIZAÇÃO E MÁQUINAS VIRTUAIS

1.1 Conceito de Virtualização

A virtualização é uma tecnologia que oferece uma camada de abstração dos verdadeiros recursos de uma máquina, provendo um *hardware* virtual para cada sistema, com o objetivo de "esconder" as características físicas e à forma como os sistemas operacionais e aplicações interagem com os recursos computacionais.

As principais qualidades da virtualização são: o reaproveitamento de recursos, a portabilidade e a segurança.

Com a virtualização podemos:

- Executar diferentes sistemas operacionais em um mesmo *hardware* simultaneamente.
- Executar um sistema operacional (e suas aplicações) como um processo de outro.
- Utilizar sistemas operacionais e aplicações escritas para uma plataforma em outra, além de outros usos que serão vistos a seguir.

1.2 Implementações

Conceitualmente a virtualização pode ser implementada de duas formas: por soluções combinadas em *hardware* e *software*, ou totalmente baseada em *software*.

As soluções de combinação entre *hardware* e *software* não são um conceito novo e suas origens remetem ao início da história dos computadores nos anos 60, nas máquinas VM/370 da IBM. Na solução de *hardware* e *software* há uma cooperação entre um

software virtualizador (que faz o papel principal) com o *hardware*, cujo qual fornece partes chaves do processo. O desempenho é a principal vantagem desta tecnologia. São exemplos de arquiteturas que suportam este tipo de virtualização: IBM z/VM e HP-UX Virtual Partition.

Na virtualização totalmente baseada em *software*, não é preciso um *hardware* provendo recursos para suportá-la, ao invés disso, é o *software* virtualizador que provê totalmente os recursos no processo. Essa tecnologia tem como vantagens o baixo custo de implementação e a portabilidade entre plataformas. São exemplos dessa tecnologia: VMWare, Xen, Microsoft Virtual Server, Solaris Zones, FreeBSD Jails e outras.

Este trabalho visa apresentar os principais conceitos da virtualização baseada por *software*, uma vez que esta tecnologia é a mais comum atualmente e vem ganhando cada vez mais popularidade por profissionais de TI (Tecnologia da Informação). A virtualização baseada em *software* além de possuir as vantagens já citadas, ainda possibilita virtualizar arquiteturas de baixo custo como, por exemplo, x86 e PowerPC.

1.3 Virtualização e emulação

É importante salientar que os termos virtualização e emulação de sistemas apesar de parecerem referir-se ao mesmo tema, na verdade possuem grandes diferenças. Um emulador é um agente escrito para tornar possível a interação entre dois sistemas distintos e incompatíveis entre si (os quais podem ser um *software* e um *hardware*, ou um *software* e outro *software*). Para isto, o emulador "traduz" as instruções entre um sistema e outro, intermediando o processo.

Já a virtualização, por sua vez, utiliza a emulação e outras técnicas para oferecer um conjunto completo de recursos, com o objetivo de permitir que vários sistemas executem sobre uma mesma plataforma visando o máximo de desempenho.

O foco da emulação é fazer um sistema executar totalmente sobre outro para o qual não foi originalmente construído, mesmo que isto cause uma perda de desempenho. De um

modo contrário, a virtualização preocupa-se sempre com o desempenho e executa o sistema virtual diretamente no *hardware* quando possível.

1.4 Máquina virtual e máquina real

Uma máquina virtual é um espaço virtual isolado com acesso ao *hardware*, onde funciona um sistema virtual. Máquina virtual (em inglês, *Virtual Machine – VM)* é o termo a que nos referimos quando estamos trabalhando com sistemas virtuais executando em uma máquina real. Uma máquina virtual é "*Uma duplicata eficiente e isolada de uma máquina real"* [POPEK e GOLDBERG].

Uma máquina real é formada por vários componentes físicos que fornecem operações para o sistema operacional e suas aplicações. "*A funcionalidade e o nível de abstração de uma máquina virtual encontram-se em uma posição intermediária entre uma máquina real e um emulador, de forma que os recursos de hardware e de controle são abstraídos e usados pelas aplicações*" [LAUREANO 2006].

O aparecimento do termo máquina virtual data do ano de 1967 quando pesquisadores da IBM desenvolveram o sistema CP-67 que permitia que um único *hardware* da família 360 simulasse múltiplas máquinas lógicas menores e totalmente independentes entre si. Entretanto este termo permaneceu pouco conhecido até a metade dos anos 90, quando o termo ressurgiu, sobretudo com o aparecimento de aplicações voltadas a virtualizar recursos nos microcomputadores PC (*Personal Computer*) de plataforma Intel (arquitetura x86).

Em ambientes virtualizados as máquinas virtuais simulam uma réplica física de uma máquina real. Dispositivos adicionais como *drives* de disquetes, CD-ROM e dispositivos USB (*Universal Serial Bus*) também podem ser compartilhados entre as máquinas virtuais e o sistema anfitrião (também chamado de sistema *host*). Os usuários têm a ilusão de que o sistema está disponível para seu uso exclusivo.

Embora as funcionalidades das diversas máquinas virtuais sejam diferentes, todos compartilham de atributos comuns [ROSEMBLUM] como:

- Compatibilidade do *software*: a máquina virtual fornece uma abstração compatível de modo que todo o *software* escrito para ela funcione.
- Isolamento: garante que os *softwares* executados em cada uma das máquinas virtuais e os da máquina real estejam totalmente isolados entre si.
- Encapsulamento: é usado para manipular e controlar a execução do *software* na máquina virtual.
- Desempenho: adicionar uma camada de *software* a um sistema pode afetar o desempenho do *software* que funciona na máquina virtual, mas os benefícios de uso de sistemas virtuais devem compensar.

1.5 Motivação para o uso máquinas virtuais

Os sistemas computacionais tradicionais, de uma forma básica, são projetados com três componentes: o *hardware*, o sistema operacional e as aplicações.

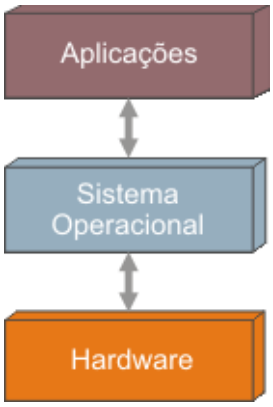


Figura 1.01 - Representação dos componentes dos sistemas tradicionais

O *hardware* executa as operações solicitadas pelas aplicações. O sistema operacional recebe as solicitações das operações por meio das chamadas de sistemas e controla o *hardware*.

Devido ao fato de os projetistas de *hardware*, sistema operacional e aplicações trabalharem independentemente, geraram ao longo dos anos várias plataformas

operacionais diferentes e incompatíveis entre si. Assim as aplicações escritas para uma plataforma não funcionam para outra.

A utilização de máquinas virtuais possibilita contornar essa dificuldade, pois permite que diferentes aplicações de diferentes plataformas executem ao mesmo tempo em um mesmo *hardware*.

Em ambiente virtualizado podemos, por exemplo, criar uma máquina virtual executando Windows, outra executando Linux, outra executando FreeBSD, além de outras, as quais executarão simultaneamente no mesmo computador. É o monitor de máquinas virtuais que permite estas implementações.

1.6 O Monitor de Máquinas Virtuais

O monitor de máquinas virtuais (*Virtual Machine Monitor* – VMM) é uma aplicação que implementa uma camada de virtualização, a qual permite que múltiplos sistemas operacionais funcionem sobre um mesmo *hardware* simultaneamente.

"As finalidades primárias de um sistema operacional são habilitar aplicações a interagir com um hardware de computador e gerenciar recursos de hardware e software de um sistema. Por tal motivo, o monitor de máquinas virtuais pode ser definido como um sistema operacional para sistemas operacionais" [LAUREANO 2006].

É o monitor de máquinas virtuais que cria e gerencia os ambientes de máquinas virtuais, interpretando e emulando o conjunto de instruções entre as máquinas virtuais e a máquina real (*hardware*).

As principais funções do monitor de máquinas virtuais são:

- Definir o ambiente de máquinas virtuais.
- Alterar o modo de execução do sistema operacional convidado de privilegiado para não privilegiado, e vice-versa.

- Emular as instruções e escalonar o uso da CPU para as máquinas virtuais.
- Gerenciar acesso aos blocos de memória e disco destinados ao funcionamento das máquinas virtuais.
- Intermediar as chamadas de sistema e controlar acesso a outros dispositivos como CD-ROM, *drives* de disquete, dispositivos de rede, dispositivos USB.

Segundo Popek e Goldberg, um monitor de máquinas virtuais deve ter três características [POPEK e GOLDBERG] principais:

- Eficiência: é extremamente importante que um grande número de instruções do processador virtual seja executada diretamente pelo processador real, sem que haja intervenção do monitor. As instruções que não puderem ser tratadas pelo processador real precisam ser tratadas pelo monitor.
- Integridade: todas as requisições aos recursos de *hardware* devem ser alocadas explicitamente pelo monitor (memória, processamento etc).
- Equivalência: o monitor deve prover um comportamento de execução semelhante ao da máquina real para o qual ele oferece suporte de virtualização, salvo haja a necessidade de se fazer alterações na disponibilidade de recursos da máquina.

Ainda segundo Popek e Goldberg, é importante salientar que na implementação de um monitor de máquinas virtuais deve-se levar em conta características como: compatibilidade, desempenho e simplicidade. A compatibilidade é importante para a execução do legado de *software*. O desempenho é de extrema importância para a execução do sistema operacional e aplicações na máquina virtual. A simplicidade é buscada para evitar falhas no monitor, que causaria problemas para todas as máquinas virtuais em execução.

1.7 Tipos de Máquinas Virtuais

Para a construção de máquinas virtuais, existem duas abordagens:

- Máquinas virtuais clássicas ou de Tipo I.
- Máquinas virtuais hospedadas ou de Tipo II.

1.7.1 Máquinas virtuais clássicas ou de Tipo I

Nesta abordagem o monitor de máquinas virtuais é implementado entre o *hardware* e os sistemas convidados (também chamados de sistemas *guest* ou *guest systems*).

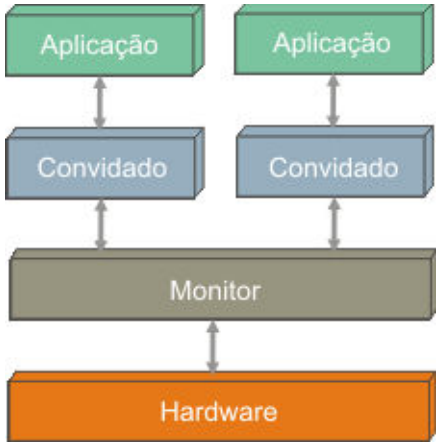


Figura 1.02 - Esquema representativo de máquinas virtuais do tipo I

O monitor possui controle sobre o *hardware* e cria um ambiente de máquinas virtuais dando a cada máquina virtual o comportamento de uma máquina física, podendo executar sobre esses ambientes, sistemas operacionais iguais ou diferentes, totalmente isolados entre si.

Um monitor deste tipo executa com a maior prioridade sobre os sistemas convidados de forma que ele pode interceptar e emular todas as operações que acessam ou manipulam os recursos de *hardware* provenientes dos sistemas convidados.

1.7.2 Máquinas virtuais Hospedadas ou de Tipo II

Nesta abordagem o monitor é implementado como um processo de um sistema operacional "real" (sistema anfitrião).

O monitor de tipo II funciona de forma análoga ao de tipo I, com a diferença que ele é executado sobre o sistema operacional anfitrião, como um processo deste. Neste modelo o monitor simula todas as operações que o sistema anfitrião controlaria.

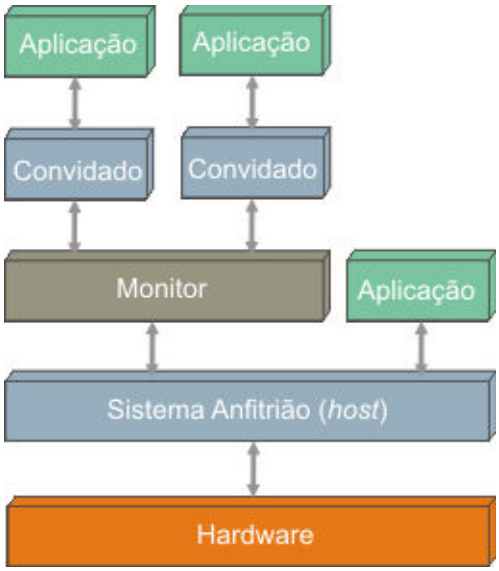


Figura 1.03 - Esquema representativo de máquinas virtuais do tipo II

1.7.3 Considerações

Deve-se considerar que os monitores de tipo I e tipo II raramente são projetados em sua forma conceitual. Hoje em dia são inseridas otimizações para aumentar o desempenho dos sistemas convidados. São elas:

Em monitores de tipo I:

- O sistema convidado acessa diretamente o *hardware*. Neste processo, é necessário que sejam feitas alterações no sistema operacional para que suporte a otimização.

Em monitores de tipo II:

- O sistema convidado acessa diretamente o sistema anfitrião. Nessa otimização, o monitor permite que sejam acessadas certas API[1] do sistema anfitrião pelo sistema convidado.
- O sistema convidado acessa diretamente o *hardware*. Essa otimização é conseguida quando o monitor permite o acesso direto a *drivers* de dispositivo do sistema anfitrião.
- O monitor acessa diretamente o *hardware*. Nessa otimização o monitor tem seus próprios *drivers* de dispositivos acessando o *hardware* com uma interface própria (de baixo nível).

1.8 Formas de virtualização por *software*

A seguir é apresentada uma visão geral sobre as principais técnicas utilizadas para conceber softwares de virtualização. Essas técnicas são apresentadas de forma mais detalhada nos capítulos IV, V, VI, VII e VIII, e também estudos sobre os principais *softwares* de virtualização que as implementam.

1.8.1 Camada de Abstração do *Hardware* (*Hardware Abstraction Layer* - HAL)

Nesta virtualização é utilizado um monitor de máquinas virtuais que simula um conjunto de recursos de *hardware* para os sistemas convidados.

Com esta técnica é possível executar diferentes sistemas operacionais (isto é, com *kernels* diferentes) em uma mesma arquitetura.

[1] API (*Application Programming Interface*) é o conjunto de rotinas que acessam ou executam determinadas funcionalidades nos sistemas operacionais. São geralmente utilizadas por programadores no desenvolvimento de *softwares* para que não seja necessário envolver-se em detalhes de implementação dos sistemas operacionais.

Exemplos: VMware, Xen, Virtual Server e VirtualBox.

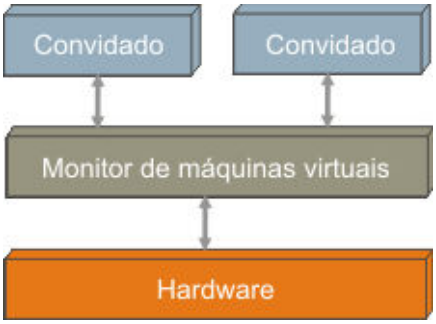


Figura 1.04 - Representação da virtualização por camada de abstração de *hardware*

1.8.2 Nível do sistema operacional (OS *Level*)

Nessa técnica, a virtualização é obtida utilizando-se uma chamada de sistema (*system call*) específica, criando um ambiente idêntico ao sistema operacional anfitrião (geralmente uma instância dele mesmo), cujo principal objetivo é o isolamento de processos. A máquina virtual funciona como um processo do sistema anfitrião. Essa forma de virtualização tem como vantagem o desempenho por já fazer parte do sistema operacional. A desvantagem é de que o usuário não pode fazer uso de outro sistema operacional (outro *kernel*) no ambiente virtual.

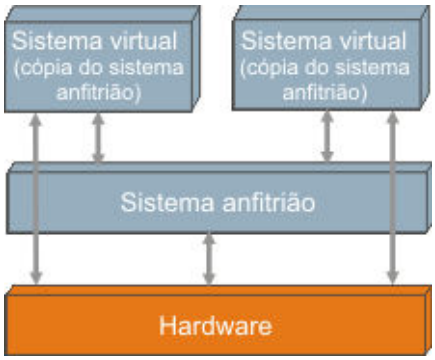


Figura 1.05 - Representação da virtualização no nível de sistema operacional

1.8.3 Emulação completa

A execução do sistema virtual é conseguida usando-se um *software* emulador, o qual simula o *hardware* do sistema para a execução do sistema convidado, "traduzindo"

instruções do sistema convidado para equivalentes no sistema anfitrião e vice-versa. Por este motivo a principal desvantagem desta implementação é o baixo desempenho. Além disso, um emulador geralmente só permite a execução de um sistema convidado por vez, sendo necessário executar uma nova instância do emulador para cada máquina virtual criada, o que contribui significativamente para a perda de desempenho.

Os emuladores são descritos neste trabalho, por se constituir uma maneira de se obter a execução de um sistema virtual. Contudo, os emuladores não são classificados como *softwares* de virtualização propriamente dito, uma vez que não preenchem os requisitos de flexibilidade e desempenho.

Exemplos: QEMU e Bochs.

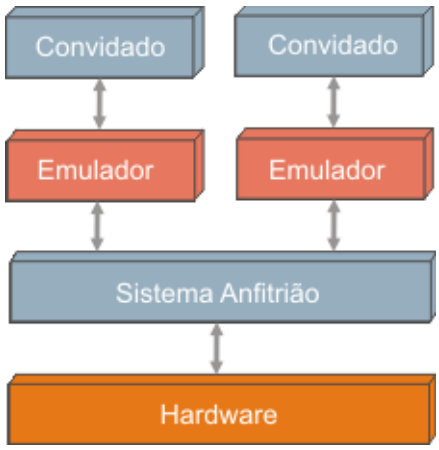


Figura 1.06 - Representação da execução de sistemas virtuais por emulação

1.8.4 Nível de Aplicação

Essa virtualização consiste no uso de uma máquina virtual como um componente-chave para execução de certas aplicações de uma forma protegida a outros processos em execução no sistema operacional. Essa máquina virtual pode ser implementada em várias plataformas, garantido assim a portabilidade da aplicação nas diversas plataformas.

Exemplos: Java VM e Microsoft .Net CLR.

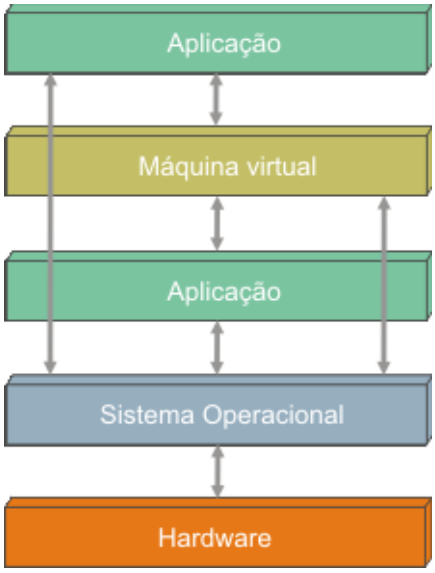


Figura 1.07 - Representação da virtualização no nível de aplicação

1.8.5 Bibliotecas de interface do usuário (*User level library interface*)

Esta virtualização consiste em prover um ambiente para execução de aplicações sobre uma plataforma diferente para o qual ela foi construída. Baseia-se na emulação das chamadas à API entre a aplicação e o sistema operacional original, traduzindo-as para equivalentes ao sistema operacional anfitrião.

Exemplo: Wine e Cygwin.

CAPÍTULO II

PRINCIPAIS APLICAÇÕES DE MÁQUINAS VIRTUAIS

2.1 Introdução

A utilização da virtualização, ao longo dos anos, tem-se revelado uma alternativa interessante em diversos paradigmas da computação, entre eles estão a centralização e consolidação de servidores, as otimizações de *hardware* e a segurança da informação.

Com a consolidação de servidores, ao invés das empresas utilizarem vários servidores com seus respectivos sistemas operacionais, utiliza-se um servidor com máquinas virtuais abrigando vários sistemas operacionais com suas aplicações e serviços, reduzindo-se assim diversos custos administrativos e operacionais.

Além disso, um sistema funcionando em uma máquina virtual fica disponível instantaneamente. Máquinas virtuais basicamente são arquivos armazenados no disco e podem ser mantidos em espera e restaurados em poucos segundos, com aplicações e serviços voltando a funcionar no mesmo ponto onde a máquina virtual foi suspensa, tornando prática sua administração.

O isolamento entre as máquinas virtuais e o sistema anfitrião torna certas tarefas arriscadas, completamente seguras. A avaliação de novos sistemas, testes com vírus e outras ameaças geralmente significa sujeitar o computador a operações onde nem sempre é possível sua reversão. Quando o teste é concluído, o estado completo da máquina virtual pode ser arquivado ou descartado. Se alguma ameaça danificar o sistema, a máquina virtual pode ser descartada e restaurada ao seu estado original.

O campo de desenvolvimento de *softwares* é outra área que também se beneficia do uso de máquinas virtuais. Desenvolver aplicações de baixo nível ou componentes de sistemas operacionais usando-se a plataforma que está em execução no momento pode

causar danos irreversíveis ao sistema. As máquinas virtuais também isolam o sistema principal de *bugs* em *softwares*, aumentando a segurança para testá-los.

A seguir serão descritas algumas das principais aplicações em que o uso da virtualização tem um papel muito importante, seja reduzindo custos ou até mesmo viabilizando certos procedimentos.

2.2 Consolidação de servidores

Hoje em dia as empresas vêm cada vez mais buscando a centralização e diminuição do número de servidores físicos em suas instalações. Esse processo é conhecido como consolidação de servidores.

Devido aos anos de recessão que estamos sofrendo recentemente, cada vez mais faz-se necessário reduzir custos. Este fato vem mudando o cenário do mercado internacional de servidores, que está caminhando novamente para centralização de recursos.

Imaginemos uma situação onde tivéssemos que disponibilizar serviços em vários servidores diferentes sendo que cada servidor teria, conforme o serviço, determinada quantidade de memória e espaço em disco. Devemos considerar também que grande parte do tempo estes servidores ficarão ociosos ou seu uso cai consideravelmente em determinados horários. Todos os servidores devem ficar disponíveis em uma rede, porém precisam ter certo nível de segurança.

Esse cenário tradicionalmente exigiria um grande investimento em infra-estrutura em computadores, espaço físico, rede, além de gastos operacionais como energia elétrica, manutenção dos equipamentos e da rede, e administração dos diversos sistemas e serviços.

Nesses casos a virtualização é uma grande alternativa, pois, ao invés de possuirmos vários servidores físicos, podemos possuir apenas alguns ou mesmo somente um, reduzindo drasticamente a complexidade. Além disso, haverá também um melhor aproveitamento dos recursos computacionais, redução do custo total de propriedade e do

custo operacional, a diminuição do consumo de energia elétrica reduzindo o impacto no meio ambiente, além do que, com o número de servidores físicos reduzidos, o espaço físico necessário para abrigá-los também ficará reduzido, garantindo vantagens como economia em administração, manutenção e refrigeração dos equipamentos.

Além disso, a capacidade de executar *softwares* de diferentes sistemas operacionais num mesmo *hardware* reduz o "desperdício" de capacidade de processamento que ocorre freqüentemente nos servidores em determinados horários ou dias do mês. Segundo a revista INFO [INFO249], "em recente estudo, somente 25% (vinte e cinco por cento) da capacidade instalada de TI é utilizada em um ano".

A administração facilitada é outro benefício agregado e de grande importância no projeto de consolidação de servidores. Nesta proposta, tecnicamente os sistemas operacionais são abrigados em máquinas virtuais e cada máquina virtual geralmente é armazenada no disco da máquina física como um só arquivo, o que torna extremamente prático, por exemplo, operações de *backup*, mudança de máquinas virtuais de um servidor físico para outro, adicionar cópias de sistemas e testar novos sistemas. Alguns monitores de máquinas virtuais (como o VMware e o Xen, que serão vistos a seguir) já possuem scripts automatizados de backup e mudança de máquina virtual de servidor.

Por fim, as empresas que consolidarem seu parque de servidores poderão aproveitar os seus equipamentos descartados para outras finalidades. Elas podem utilizar estes equipamentos, por exemplo, para aumentar a disponibilidade e segurança de seus sistemas, implantando soluções de tolerância à falhas.

2.2.1 Implementação da consolidação de servidores

Como já foi visto, a consolidação de servidores é uma excelente alternativa para redução de custos em ambiente de TI. Na virtualização um servidor é "subdividido" em máquinas virtuais e cada máquina virtual tem sua própria CPU virtualizada, memória, e espaço em disco e podendo compartilhar outros dispositivos anexados à plataforma física.

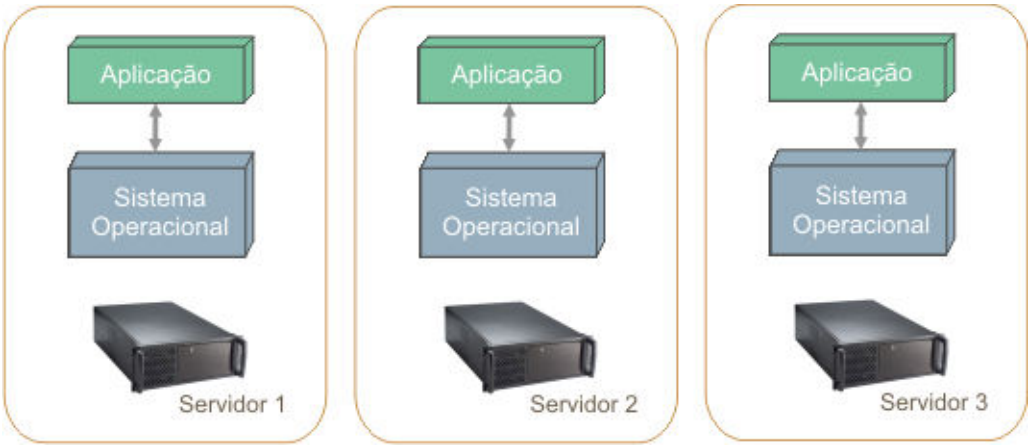


Figura 2.01 - Servidores não virtualizados

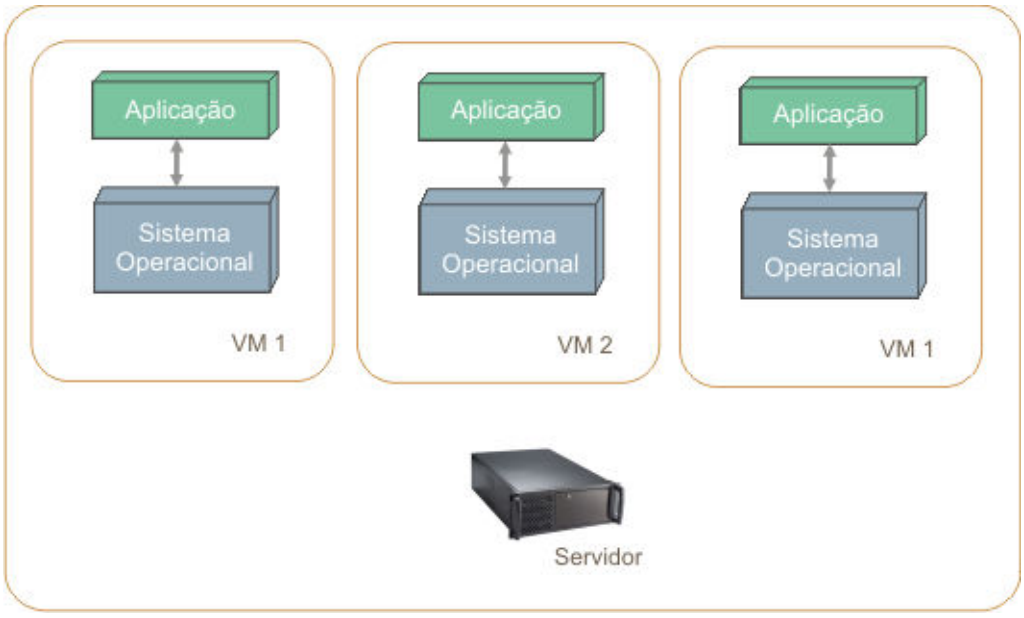


Figura 2.02 - Servidor virtualizado

Para implementarmos uma consolidação de servidores, primeiramente devemos considerar alguns pontos chaves [INTEL]:

- Avaliar os processos e as aplicações críticas. Onde aumentando o nível dos serviços, aumentariam os resultados?
- Verificar quais servidores são subutilizados e que poderiam compartilhar recursos. Muitos servidores *web* e de e-mails hoje funcionam na maioria do tempo abaixo de 10% de utilização. Esses são ótimos candidatos a uma consolidação.

- Algumas aplicações são más candidatas à consolidação. Nesta classificação estão enquadradas as aplicações de alta *performance* que utilizam o servidor na maior parte do tempo (ex: analisadores de grandes massas de dados), bem como, os de missão crítica ou de *performance* crítica, em que qualquer contenção de recursos poderia impactar negativamente nos resultados.

Deve-se proceder a seguinte metodologia para cálculo de capacidade:

- Fazer um histórico dos dados das aplicações a serem consolidadas, capturando dados como consultas, uso de memória, entrada e saída, armazenamento ou qualquer outra informação relevante, em um ciclo de um dia, um mês, um ano etc.
- Usando este histórico, deve-se mapear padrões de uso para descobrir aplicações que poderiam funcionar juntas, e combinando aplicações de picos em horários diferentes.
- Somar todos os picos de carga de trabalho de todos os serviços e aplicações a serem consolidadas para determinar a requisitos de capacidade computacionais necessárias (CPU, memória, disco, operações de entrada e saída etc.).
- Acrescentar uma margem de crescimento projetada para os próximos seis meses, dois anos ou eventual substituição de equipamentos (por exemplo: devido ao término de suas vidas úteis).
- Caso haja políticas institucionais de utilização de servidores (por exemplo: nenhum servidor deveria estar funcionando normalmente acima de 80% da capacidade), deve-se adicionar também esta margem de segurança.
- Por fim, deve-se considerar no cálculo o *overhead*[2] gerado da técnica de virtualização adotada.

Após este estudo, deve-se realizar a consolidação das aplicações em sua forma prática, consolidando aplicações de menor escala e, na medida em que se obtém êxito, proceder para aplicações de maior escala.

[2] *Overhead*: Custos adicionais em processamento ou armazenamento que reduz de forma significativa e indesejável, o desempenho dos sistemas de computação.

Ao final de cada trabalho, deve-se utilizar técnicas de medição de carga de trabalho para uma análise do sucesso obtido.

2.3 Segurança

Ameaças à segurança são uma das mais imprevisíveis e desafiadoras causas de quedas de serviço em sistemas, o que pode ser extremamente custoso para a instituição. Em recente relatório apontando pelos laboratórios McAfee Avert, as taxas em que novas ameaças à segurança aparecem duplicaram nos últimos anos [HINES]. O relatório semestral da Symantec sobre ameaças de segurança, referente ao segundo semestre de 2005, apontou aumento no número ameaças, enfatizou o uso de novas e sofisticadas ferramentas para a prática do crime cibernético, e o substancial aumento no roubo de informações confidenciais [SYMANTEC].

A virtualização é uma poderosa ferramenta na prevenção contra ameaças de segurança, na medida em que evita custos com interrupção de serviço e perda de dados.

2.3.1 Isolamento

Um dos maiores benefícios da virtualização é o isolamento de processos em máquinas virtuais. Com a virtualização, podemos isolar aplicações de alto risco de aplicações potencialmente vulneráveis. O isolamento eleva a proteção contra aplicações maliciosas, aumentando a dificuldade das mesmas acessarem dados, ou afetar processos que estejam executando em outras máquinas virtuais.

Além disso, isolando falhas, prevenimos que uma aplicação em mau funcionamento comprometa todo o sistema. Por exemplo, aplicações que fazem conexões externas, como servidores de e-mail, podem ser confinadas em máquinas virtuais próprias, ou seja, mantendo-as separadas do sistema operacional anfitrião. Além do mais, podemos definir nesta máquina virtual, privilégios para acesso a dados e serviços vitais.

Ainda neste contexto, uma máquina virtual pode ser usada para dar privilégios limitados a usuários ou aplicações convidados em um ambiente que pode ser seguramente "deletado" quando o serviço for concluído. Se algumas destas máquinas virtuais estiverem comprometidas, somente seria necessário descartar as máquinas virtuais corrompidas e iniciar novas outras, ou recuperar cópias de seguranças (*backup)*.

2.3.2 Tolerância à falhas

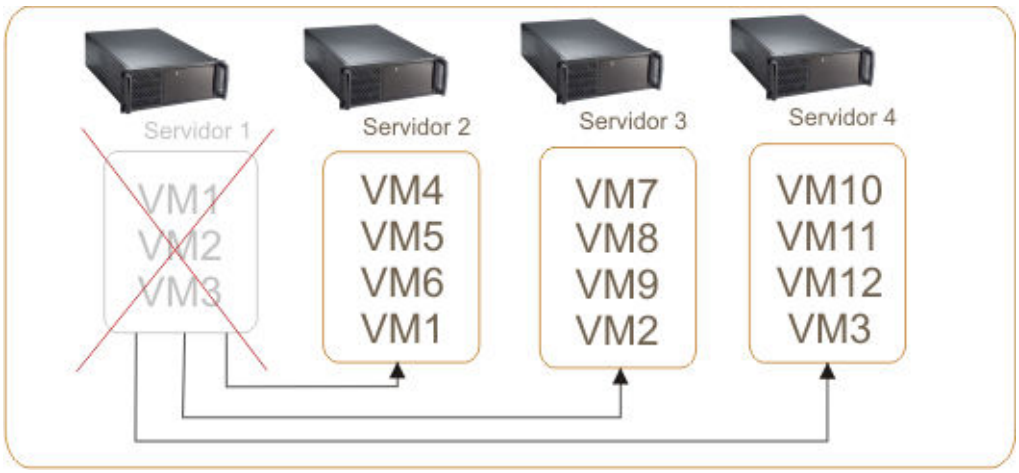
A virtualização permite que uma aplicação em mau funcionamento executando em sua máquina virtual possa ser reiniciada rapidamente e de um modo seguro, sem afetar outras aplicações que estejam executando no mesmo sistema anfitrião. Ela permite também, que administradores possam suspender, reiniciar e migrar máquinas virtuais, ajudando a suavizar efeitos de ataques e falhas em aplicações, ou até mesmo falhas em *hardware*, quando acontecerem (figura 2.03).

Em caso de falha, as máquinas virtuais podem simplesmente ser reiniciadas de um ponto de restauração, uma cópia de segurança, ou outro mecanismo de recuperação que possa trazer o sistema a um estado saudável.

Se a camada física do sistema falha, as máquinas virtuais podem ser migradas e restauradas em outras máquinas físicas. Esta habilidade para recriar um serviço sem ter que "iniciar do zero" é especialmente útil para serviços de missão crítica e de execução prolongada, porque provê uma rápida recuperação de desastres. Uma boa idéia é armazenar pontos de restauração de máquinas virtuais em locais remotos para posteriormente serem recuperados em caso de falha.

Os monitores de máquinas virtuais VMware e Xen dispõem de ferramentas que migram e reiniciam as máquinas virtuais automaticamente em outros servidores em caso de falha do servidor original.

Funcionamento normal



Funcionamento com falha

Figura 2.03 - Demonstração de recuperações em uma falha de sistema

2.3.3 *Honeypots* e *Honeynets*

Um *honeypot* é um sistema ou aplicação que é colocado em uma rede de forma proposital para que seja comprometido ou "atacado". Um *honeypot* age como uma ferramenta de estudos e testes de vulnerabilidades em uma rede ou aplicações, mas não é considerado uma ferramenta de segurança. *Honeynet* é uma rede composta de *honeypots*.

Máquinas com *honeypots* instalados executam serviços falsos, que um atacante trata como se fosse um serviço original. Entretanto serviços como este têm a finalidade de monitorar os passos do atacante para que se saiba como e quando foram feitos os

ataques e o que se pretende fazer após a invasão do sistema, gerando *logs* dessas informações para os administradores do sistema.

Os *honeypots* não têm valor de produção, assim qualquer um que tente se comunicar com o sistema é geralmente uma tentativa de ataque.

A virtualização faz o uso de *honeypots* e *honeynets* ser uma alternativa extremamente segura, simples e barata. Por exemplo, com a virtualização podemos simular uma rede virtual colocando *firewall*, *software* IDS, servidor *web* e servidor de arquivos em máquinas virtuais isoladas executando sobre a mesma máquina física. Um atacante que consiga invadir este sistema acreditará que está invadindo uma rede real, e assim, uma vez que toda a ação do atacante foi devidamente registrada no *logs* do sistema, podemos identificar as vulnerabilidades exploradas por ele e eliminá-las de nossa rede de produção.

2.3.4 Detecção de intrusão

Sistemas de detecção de intrusão (*Intrusion Detection System* – IDS) são utilizados para detectar se alguém está tentando invadir um sistema ou fazer mau uso dele. Essas ferramentas foram concebidas com o crescimento das redes, devido à dificuldade de administradores de redes e analistas de segurança monitorar ou inspecionar constantemente todos os sistemas e arquivos de *logs*.

As ferramentas IDS têm a finalidade de ficar a todo tempo monitorando um sistema e tentando reconhecer ações intrusivas ou fora de um padrão normal de uso. Caso detectem alguma anomalia, elas automaticamente disparam um alerta ao administrador do sistema ou executam ações defensivas.

Existem conceitualmente dois tipos de implementação de ferramentas IDS:

- *Host Based IDS* (HIDS): são instalados em um servidor (*host*) para prevenir ataques ao próprio servidor. Essa implementação é utilizada quando se deseja elevar a segurança das informações contidas neste servidor.

- *Network Based IDS* (NIDS): essa implementação baseia-se na captura e análise dos pacotes na rede, e são instalados em máquinas em que se concentra o tráfego de rede.

O uso de máquinas virtuais com ferramentas IDS tem mais sentido quando as implementamos em um sistema anfitrião devido a um problema característico dos sistemas HIDS: se as ferramentas IDS estiverem instaladas na própria máquina que está sendo monitorada, o sistema fica passível de desativação por parte do atacante, o que causa redução de sua confiabilidade.

A idéia deste modelo consiste em instalar as ferramentas IDS num sistema operacional anfitrião e confinar o sistema a ser monitorado em uma máquina virtual executando sobre esse sistema anfitrião. Dessa forma o sistema principal ficaria isolado das ferramentas IDS e este por sua vez estaria fora do alcance do atacante, monitorando constantemente os acessos à máquina virtual.

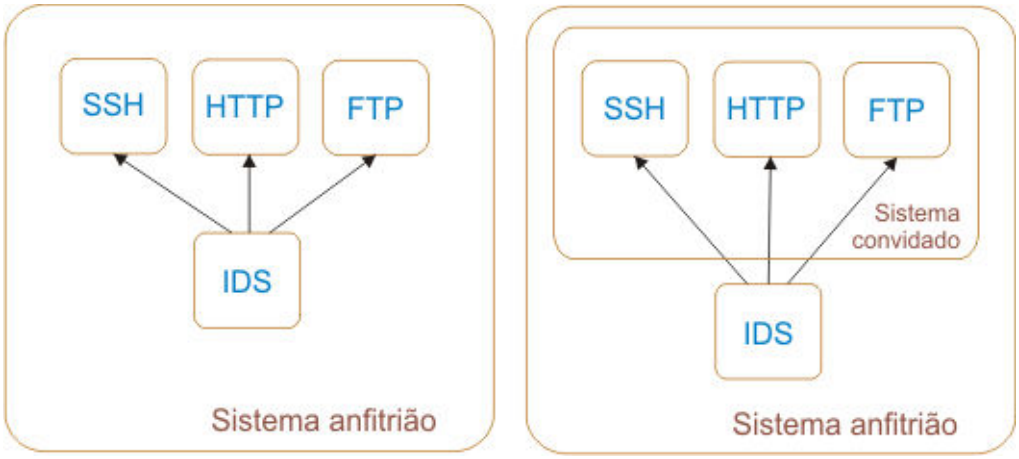


Figura 2.04 - Representação de um modelo freqüentemente utilizado e um modelo isolado em máquinas virtuais

A figura 2.04 ilustra, no primeiro caso, uma implementação normal. Nessa implementação o atacante que chegar ao sistema pode ter acesso às ferramentas IDS e conseqüentemente desativá-las. No segundo as ferramentas IDS estão isoladas e monitorando o sistema convidado, um atacante que chegar ao sistema convidado não conseguirá chegar até as ferramentas IDS.

2.4 Ensino

A força da virtualização também pode ter excelente emprego na área de ensino. O uso de máquinas virtuais torna o processo de ensino bastante prático e a custo reduzido. Por exemplo, em uma instituição que ministra diversos tipos de cursos e que utiliza a tecnologia dos computadores, é comum a necessidade de possuir máquinas dedicadas a cada curso, o que gera grandes custos de compra de equipamentos e manutenção. Também, nestes casos, é comum acontecerem diversos problemas durante o curso, tais como: necessidade de reinstalação do sistema operacional e aplicativos, dificuldade de se restaurar configurações padrão, alteração e esquecimento de senhas (como a de administrador), arquivos apagados de forma acidental, etc.

Problemas como estes podem ser facilmente resolvidos com o uso dos conceitos de virtualização. Para essa implementação podem ser consideradas duas abordagens:

Na primeira abordagem pode ser utilizado um computador de grande porte como servidor central, onde ficarão armazenadas as máquinas virtuais que serão utilizadas pelos alunos dos diversos cursos. No lado do aluno serão utilizadas máquinas clientes de pequeno porte, onde acessarão, através de uma conexão de rede, as máquinas virtuais armazenadas no servidor. Periodicamente os arquivos contendo as máquinas virtuais originais poderiam ser recuperados no servidor, o que rapidamente tornaria o sistema pronto para um novo uso.

Na segunda abordagem pode ser utilizado um número reduzido de computadores contendo várias máquinas virtuais com o conteúdo dos cursos. Nesta abordagem cada aluno utilizaria o computador podendo fazer suas modificações conforme necessário, todavia restrito a seu espaço virtual. Ao final do período, podem ser recuperados os arquivos de máquinas virtuais originais alojados em uma área de armazenamento remoto, para que rapidamente os sistemas retomem seu estado normal.

Em ambas as abordagens os alunos devem armazenar seus arquivos de curso e configurações pessoais em sua mídia de preferência ou em um espaço em disco remoto, uma vez que as máquinas virtuais sempre serão restauradas em seu estado original.

Sabemos que é comum em processos de aprendizagem acontecer danos ao sistema operacional ou ferramentas instaladas, seja para ensino de computação, seja para ensino em outras áreas. Estas abordagens têm como principal vantagem o fato de se poder criar máquinas virtuais instalando o sistema operacional e ferramentas que sejam necessárias para cada perfil de curso/aluno, em um estado em que todo o sistema esteja testado e livre de problemas, e, posteriormente permitindo que se recupere essas configurações, sem maiores transtornos aos alunos e professores.

2.5 Teste e Migração de aplicações

Sem dúvida um dos grandes desafios para profissionais de TI é o teste e a migração de aplicações. Com o passar do tempo é necessário que as aplicações utilizadas sejam atualizadas para que possamos "desfrutar" de novos recursos e tecnologias existentes. Mas por outro lado essa não é uma tarefa tão simples, uma vez que ao substituirmos uma aplicação por outra, ou atualizarmos para uma versão mais nova, corremos o risco de ficarmos impossibilitados de utilizar estas aplicações, ocasionando prejuízos para a instituição.

Utilizando o recurso da virtualização podemos criar máquinas virtuais para fazer os devidos testes de sistemas operacionais, migrações de aplicações, desenvolvimento e teste de *software* novo etc, tudo isso de forma isolada do sistema de produção. Assim se algo der errado podemos restaurar uma máquina virtual original e retomar o processo do início, e com isso conhecer as dificuldades do processo de forma antecipada.

2.6 Consolidação de aplicações legadas

Hoje em dia grande parte das instituições ainda utiliza aplicações legadas em seus sistemas de missão crítica, ou seja, aplicações que não podem ser migradas ou modificadas devido a fatores como: falta de orçamento, fabricante que se retirou do mercado, aplicações que foram descontinuadas pelo fabricante, etc.

Podemos utilizar a virtualização para mover estas aplicações legadas e sem suporte por parte do fabricante de um modo em que possuam um melhor aproveitamento de recursos de *hardware* e gerenciamento.

A virtualização simplifica a migração de aplicações legadas em novas plataformas. Em casos em que a nova plataforma não suporta a execução das aplicações legadas, a aplicação pode ser hospedada com seu sistema operacional (para qual foi desenvolvido originalmente) em uma máquina virtual executando sobre a nova plataforma, sem nenhuma necessidade de alteração no *software*.

Normalmente um *hardware* novo fica subutilizado quando é dedicado a executar aplicações legadas. A consolidação das aplicações legadas através da virtualização também oferece um melhor aproveitamento de *hardware* novo, pois permite que um único *hardware* execute várias instâncias de aplicações legadas simultaneamente.

Por fim, movendo aplicações legadas para sistemas virtuais eliminamos os riscos de problemas de compatibilidade e de execução simultânea, porque elas estarão confinadas às suas respectivas máquinas virtuais, completamente isoladas das novas aplicações e dos novos sistemas operacionais.

2.7 Serviço de Hospedagem (*Hosting)*

A hospedagem (*hosting*) é um serviço que possibilita que pessoas e empresas possam armazenar informações, imagens, vídeos, ou qualquer conteúdo acessível pela *web*. Esse serviço é prestado por empresas que possuem um centro de dados, e normalmente também podem oferecer conectividade à Internet.

Os provedores deste serviço geralmente usam uma técnica conhecida como *virtual hosting* para abrigar centenas de páginas com seus respectivos domínios em um servidor *web*, uma vez que seria proibitivo em termos de *hardware,* manter um servidor *web* separado para cada domínio. Através da técnica, um único servidor *web* pode atender solicitações a diversos domínios simultaneamente (figura 2.05).

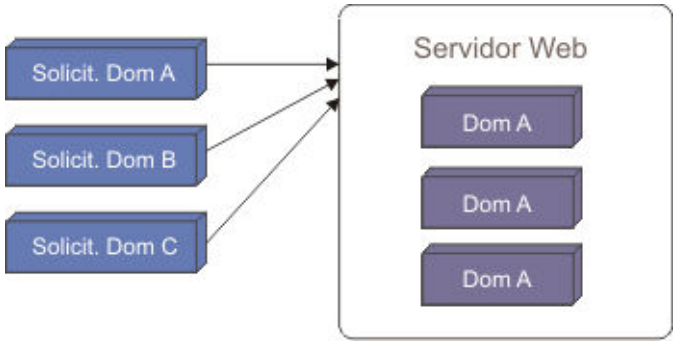


Figura 2.05 - Um servidor com vários domínios virtuais

Por outro lado, um servidor dedicado tem como principal vantagem o fato de não interferir em outros *site*s hospedados no mesmo servidor físico e liberar o acesso do sistema operacional (acesso de *root*) para que o administrador do *site* possa instalar e configurar quaisquer aplicativos com total liberdade e flexibilidade.

As máquinas virtuais possibilitam que os serviços de hospedagem possam usufruir dos benefícios citados no parágrafo anterior sem maiores custos. Essa técnica chamada de servidor virtual privativo (*virtual private server* - *VPS*), permite que os centros de dados operem com diversos servidores virtuais dedicados para cada domínio consolidados em um número reduzido de servidores físicos (figura 2.06).

Como um servidor virtual privativo executa seu próprio sistema operacional, os provedores poderão dar a seus clientes acesso com segurança às máquinas (virtuais), ampliando sua gama de serviços oferecidos.

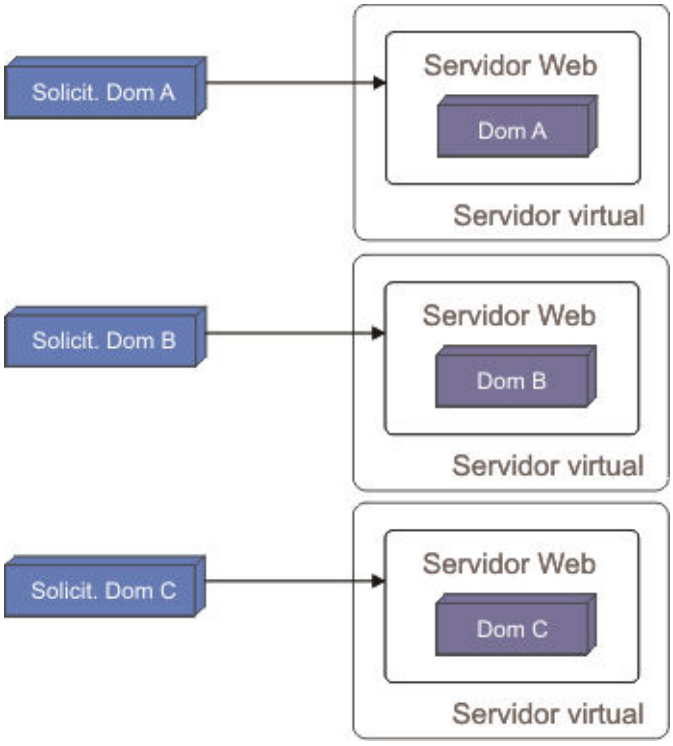


Figura 2.06 - Virtualização de servidores *web* dedicados para cada domínio

2.8 Suporte Técnico

A virtualização também facilita o trabalho de empresas que oferecem serviços de suporte técnico. Com ela as limitações técnicas que inviabilizam o atendimento de suporte para diferentes sistemas operacionais e aplicações e suas diferentes versões, praticamente são eliminados, uma vez que não há necessidade de se manter equipamentos dedicados a cada versão de sistema operacional ou aplicação que se deseja dar suporte. Além disso, ao invés de uma empresa manter funcionários específicos para atender solicitações de suporte para cada tipo de sistema operacional e aplicações, ela pode habilitar seus funcionários a atenderem chamadas de suporte aos diversos tipos de sistemas diferentes, o que acarreta em economia em mão-de-obra e qualidade no atendimento.

CAPÍTULO III

TÉCNICAS PARA CONSTRUÇÃO DE *SOFTWARE* VIRTUALIZADOR

3.1 Modos de CPU, *trap* e Chamadas de Sistema

Basicamente existem dois modos de privilégios na CPU para execução de código binário: o modo de usuário (*user mode*) e modo supervisor (*kernel mode*). O sistema operacional executa em modo supervisor, gerenciando os processos e, portanto tem o controle total da máquina. Os processos dos sistemas operacionais executam em modo usuário e por isso não detêm o controle total da máquina.

As instruções que mudam o completo estado do sistema e são chamadas de instruções privilegiadas, não podem executar no modo usuário. Quando uma aplicação em modo usuário tenta executar instruções privilegiadas, é gerada uma exceção que resulta em uma interrupção da CPU, esta exceção é comumente chamada de "*trap"***.** Em seguida, a CPU automaticamente transfere o fluxo de controle para o *kernel* do sistema operacional, o qual decide o que fazer.

Geralmente as aplicações (que executam em modo de usuário) evitam executar instruções privilegiadas. Ao invés disso, elas chamam o *kernel* através de uma funcionalidade conhecida como chamadas de sistema (*system calls*) [STEIL].

3.2 O Modo Hipervisor

Algumas CPU introduzem um terceiro modo de CPU, chamado modo hipervisor (*hypervisor mode*). O modo hipervisor tem total controle da CPU, e é onde o monitor de máquinas virtuais gerencia os sistemas operacionais abrigados em suas máquinas virtuais. Enquanto a interface entre modo usuário e modo supervisor ainda é a mesma, agora há uma nova interface entre o modo supervisor e o *hardware*, é o modo hipervisor. Desse modo, todas as instruções privilegiadas feitas pelo *kernel* são

capturadas pelo monitor de máquinas virtuais que pode emular o comportamento desejado. Este método é chamado captura e emulação (*trap-and-emulate*) [STEIL].

Um exemplo de uma CPU que implementa um modo hipervisor é o IBM PowerPC 970, também conhecido como G5. As novas arquiteturas Intel VT da Intel e AMD-V da AMD também implementam uma espécie de modo hipervisor e são descritas mais a seguir.

3.3 O problema da Arquitetura x86

Os processadores de arquitetura x86 (ou IA32) possuem quatro níveis de privilégio para execução de códigos que são numerados de 0 a 3. Código rodando no nível 0 (modo supervisor) pode executar qualquer instrução na CPU, enquanto no nível 3 (modo usuário) existem instruções que não podem ser executadas. Esses níveis de privilégios são comumente chamados de "anéis" (*rings*), devido à forma como eram ilustrados no manual de programação do chip 80386 (figura 3.01).

Figura 3.01 - Estrutura da Arquitetura x86

Nessa arquitetura os sistemas operacionais somente usam os níveis 0 e 3, sendo o nível 0 para o *kernel* e o nível 3 para o modo usuário.

Como nessa arquitetura não foi implementado um modo hipervisor, o monitor de máquinas virtuais é obrigado a executar em modo supervisor, e ao criar uma máquina

virtual ele precisa forçar o *kernel* do sistema operacional convidado a executar em modo usuário no *ring* 3 (em algumas implementações ele executa em um nível não utilizado como o *ring* 1).

Com isso, para manter cada sistema convidado isolado é necessário utilizar uma técnica complexa chamada de "desprivilegiamento", a qual força a execução de um sistema convidado em um nível menos privilegiado.

A seguir estão descritas as principais técnicas utilizadas para superar esta dificuldade da arquitetura x86. É importante salientar que tais técnicas também podem ser implementadas em outras arquiteturas.

3.4 Virtualização Total (*Full Virtualization*)

A virtualização total é uma técnica que provê uma completa simulação da subcamada de *hardware* para os sistemas convidados. O resultado é um ambiente em que todos os sistemas operacionais que são capazes de executar diretamente em um *hardware* também podem executar em uma máquina virtual.

A principal vantagem da virtualização total é que não há necessidade de modificações nos sistemas operacionais convidados para que suportem a virtualização, dado que uma estrutura completa de *hardware* é virtualizada o que faz com que o sistema convidado "pense" estar executando diretamente no *hardware.*

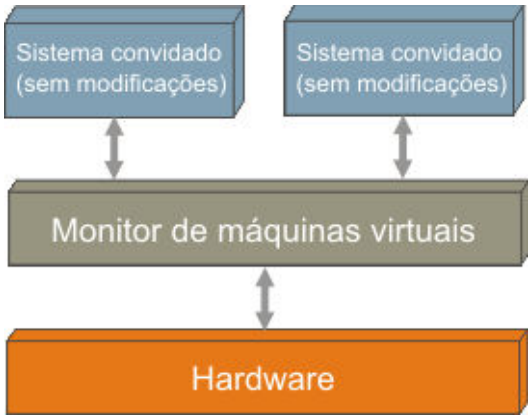


Figura 3.02 - Estrutura representativa da virtualização total

O desafio para a virtualização total é manter cada sistema convidado isolado. Para conseguir isto, são aplicadas algumas técnicas de captura e emulação de instruções da CPU, o que resulta em um desempenho reduzido neste tipo de técnica.

3.4.1 Virtualização de CPU na virtualização total

Para que uma arquitetura seja estritamente virtualizável, é necessário que todas as instruções privilegiadas sejam capturadas para o modo supervisor quando originadas em modo usuário. Entretanto em arquiteturas como a x86, o *kernel* do sistema convidado não pode estar executando em modo usuário, porque ele pode utilizar instruções *assembler* que não podem ser capturadas e emuladas. O *kernel* comporta-se diferentemente quando em modo usuário. Nesta arquitetura existem diversas instruções que tem comportamentos diferentes no modo usuário e no modo supervisor, e por isso não causam uma exceção na CPU (*trap*). Um exemplo disso é que uma aplicação em modo usuário poderia perguntar se ela está executando em modo supervisor ou em modo usuário, e obteria a mesma resposta: “modo usuário”, sem qualquer chance do monitor de máquinas virtuais interceptar essa instrução e retornar uma resposta falsa [STEIL].

Com isso dizemos que a arquitetura x86 não é estritamente virtualizável, pois não possui nativamente um modo hipervisor e nem todas as instruções sensíveis causam *traps* de CPU.

Para contornar esse problema, em algumas soluções o código x86 é analisado instrução por instrução e emulado, contudo este processo causa uma perda de desempenho considerável. Esta técnica também é conhecida como emulação do código x86 e é utilizada pelo *software* de virtualização Bochs.

Em outras soluções é utilizada uma técnica chamada tradução binária (ou reescrita binária) que é bem mais eficiente em termos de desempenho. Essa técnica atualmente é a mais utilizada e são implementados em *softwares* como VMware, Microsoft Virtual Server e outros.

A tradução binária consiste em recompilar todo e somente o código sensível, isto é, traduzir todo o código *assembler* que é problemático (que necessita de um *trap* de CPU, mas como foi explicado anteriormente, não causa um *trap*) substituindo com *traps* explícitos no modo supervisor. Todo código do modo supervisor gerado nos sistemas convidados deve ser analisado e emulado antes de ser executado.

O código é dividido em blocos e estes são então verificados. Se eles não contêm instruções problemáticas, podem ser executados, do contrário estas instruções são substituídas por outras, adequadas pelo monitor, o que fará com que o sistema convidado acredite estar executando em um modo privilegiado quando na verdade está em um modo "desprivilegiado".

Nesse contexto, todo código que já foi checado uma vez não precisa ser checado novamente, e uma definição de blocos que se refere a outro, pode ser colocado junto a um bloco maior (*trace cache*), podendo ser executado sem futuras checagens, o que agiliza bastante o processo.

3.4.2 Virtualização de memória/disco na virtualização total

Uma vez que a máquina virtual deve se comportar como uma máquina real, a memória deve se comportar da mesma forma para todo código executado dentro da máquina virtual. Entretanto, o acesso à página de memória pelo endereçamento normal de dentro de uma máquina virtual não é possível, pois todo sistema operacional acessa a página física começando com o endereço 0, o que causaria um grande conflito já que eles mapeariam suas páginas virtuais para a mesma página física.

Então cada acesso à tabela de página na máquina virtual causa um *trap,* cujo controle é transferido ao monitor de máquinas virtuais. Este por sua vez, reserva um mapeamento que possua o mesmo efeito, porém usa uma página com endereçamentos diferentes. O sistema convidado "acredita" estar usando o início da memória (endereço 0).

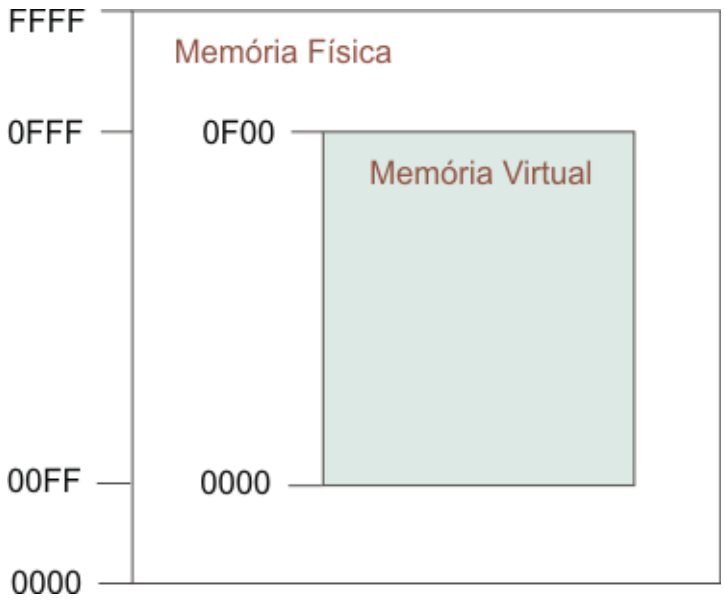


Figura 3.03 - Alocação de página de memória para a máquina virtual

Conforme a figura 3.03, a máquina física possui um endereçamento de memória iniciando em 0000 e vai até FFFF, porém o monitor de máquinas virtuais reserva o espaço para a página de memória da máquina virtual, que começa no endereço físico de 00FF a 0FFF. Contudo o sistema convidado "acredita" que esta página está começando no endereço 0000 e finalizando em 0F00. É o monitor de máquinas virtuais que se encarrega de "traduzir" os endereços corretamente.

Praticamente o mesmo processo ocorre na virtualização do disco. O espaço em disco para a máquina virtual é alocado previamente e o mapeamento é feito de forma similar ao mapeamento de memória. O sistema convidado "acredita" ter acesso exclusivo ao disco.

3.4.3 Virtualização de E/S na virtualização total

Na virtualização total, todo o código que esteja dentro de uma máquina virtual executa em modo usuário e todo acesso de Entrada e Saída (E/S) vai gerar um *trap,* direcionando o controle para o monitor de máquinas virtuais. A partir daí, o monitor "descobre" a finalidade do acesso e emula uma cópia do *hardware* retornando esta cópia ao sistema convidado. Por exemplo, quando um sistema convidado "pergunta"

pelo estado do mouse, isto gera um *trap* o qual a CPU direciona para o monitor, este por sua vez informará o estado do mouse atual para o sistema convidado baseado em informações internas sobre o mouse emulado. Quando um dispositivo virtual gera interrupções, o monitor injeta uma interrupção na máquina virtual emulando o que aconteceria se caso a interrupção fosse feita diretamente à máquina física.

Problematicamente, este método é muito lento para muito dispositivos (especialmente vídeo) porque o *driver* do sistema convidado e o monitor têm que se comunicar na linguagem de protocolo de *hardware*, que é muito eficiente para máquina real, todavia não é eficiente entre dois *softwares*. Felizmente os fabricantes de soluções de virtualização total, desenvolveram *drivers* especiais (que utilizam recursos de E/S que não estão sendo utilizados pela máquina real) para que os sistemas convidados diretamente se comuniquem com a máquina real, tornando-se um protocolo muito mais eficiente [STEIL].

3.5 Paravirtualização (*Paravirtualization)*

Paravirtualização é uma técnica que apresenta uma interface de *software* para máquinas virtuais que é similar (mas não idêntica) à subcamada de *hardware*. A técnica permite que o sistema convidado acesse diretamente recursos do *hardware*, porém com restrições, que são administradas pelo monitor de máquinas virtuais. Esta capacidade minimiza o o*verhead* e otimiza o desempenho do sistema para suportar a virtualização.

A principal limitação da paravirtualização é a necessidade de que o sistema operacional convidado seja previamente adaptado (modificado) para executar no topo de um monitor de máquinas virtuais. Entretanto, a paravirtualização elimina a necessidade da dependência dos mecanismos de *trap* da CPU*,* não havendo necessidades de capturar e emular a maioria das instruções.

A figura 3.04 representa a paravirtualização, em que o sistema convidado pode acessar diretamente o *hardware*, sendo que este processo é totalmente gerenciado pelo monitor de máquinas virtuais. Esta técnica é utilizada pelo monitor Xen, por exemplo.

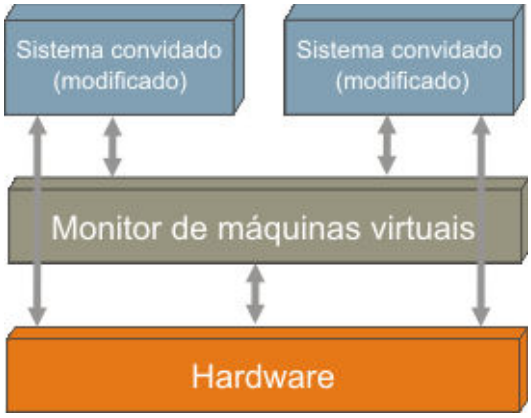


Figura 3.04 - Estrutura representativa da paravirtualização

3.5.1 Paravirtualização e CPU

A paravirtualização simplifica a interface exportada para o *hardware*, de um modo que elimina certas necessidades implementadas na virtualização total. Um exemplo são as instruções sensíveis (que executam de modos diferentes dependendo do modo que são executado, em modo supervisor ou modo usuário). Na paravirtualização elas são executadas diretamente na CPU, todavia há um pequeno número de instruções que devem ser substituídas (capturadas e emuladas).

3.5.2 Paravirtualização e memória/disco

Na paravirtualização o sistema operacional convidado (que foi modificado para suportar a virtualização) recebe do monitor de máquinas virtuais, o espaço que será utilizado para sua página de memória, e este, acessa diretamente a memória física sem que seja necessária qualquer intermediação do monitor.

O acesso ao disco também é feito de forma direta pelo sistema convidado, porém uma vez que há vários sistemas convidados executando de forma simultânea e o disco é compartilhado, o monitor de máquinas virtuais gerencia a fila de acessos [MAGENHEIMER e CHRISTIAN].

3.6 Emulação com Recompilação Dinâmica

Outra técnica bastante utilizada é a emulação com recompilação dinâmica (também conhecida como *dynamic recompilation* ou *dynarec*). Esta técnica consiste em recompilar partes do código de um programa durante sua execução. Com isso, o sistema pode adequar o código compilado para refletir o ambiente de execução original do programa, e talvez produzir um código mais eficiente, explorando informações que não estão disponíveis para um compilador estático tradicional. Em outros casos, um sistema pode empregar recompilação dinâmica como parte de uma estratégia de otimização adaptável para executar uma representação portátil do programa como os *bytecodes* do Java ou do .NET CLR [LAUREANO 2004].

Exemplificando de um modo simples a recompilação dinâmica, podemos supor que um determinado programa está sendo executado num emulador e precisa copiar uma *string* nula. O programa está compilado originalmente para um processador simples. Esse processador pode copiar apenas um *byte* por vez, e deve fazê-lo lendo a *string* de origem em um registrador, e escrevendo a *string* de destino a partir deste registrador. Esse programa se pareceria como:

```
beginning:
    mov A,[ponteiro para primeira string]
                ; Coloca o endereço do primeiro caracter da string de
                ; origem no registrador A

    mov B,[ponteiro para segunda string]
                ; Coloca o endereço do primeiro caracter da string de
                ; destino no registrador B

loop:
    mov C,[A]     ; Copia o byte do registrador A para o C
    mov [B],C     ; Copia o byte no registrador C para o B
    cmp C,#0      ; Compara o dado copiado com 0 (marcador de fim de string)
    inc A         ; Incrementa o registrador A para apontar para
                  ; o próximo byte
    inc B         ; Incrementa registrador B para apontar para
                  ; o próximo byte
    jnz loop      ; Se ele não for 0 então volta e copia o próximo byte
end:
```

Imaginemos que um emulador estivesse executando num processador que é similar a esse, todavia possuísse um desempenho melhor em cópias de *strings*, e que este emulador soubesse desta característica. Ele poderia identificar as instruções de seqüência de cópia da string e decidir reescrevê-las de um modo mais eficiente antes da execução.

Esse novo processador possui uma instrução chamada "movs" especificamente designado para copiar *strings* de uma forma mais eficiente. Esta hipotética instrução copiaria 16 bytes por vez sem ter que carregá-lo no registrador C, mas pararia quando copiasse um byte 0 (que marca o fim da string). Ele também sabe que o endereço da string estará no registrador A e B, então ele incrementa A e B em 16 bytes, cada vez que executa.

Então, o novo código compilado se pareceria com o seguinte:

```
beginning:
    mov A,[ponteiro para primeira string]
                ; Coloca o endereço do primeiro caracter da string de
                ; origem no registrador A
    mov B,[ponteiro para segunda string]
                ; Coloca o endereço do primeiro caracter da string de
                ; destino no registrador B

loop:
    movs [B],[A]  ; Copia 16 bytes do registrador A para o
                  ; registrador B, e incrementa A e B em 16
    jnz loop      ; Se não for  zero, não alcançou
                  ; o final da string, então volte e continue copiando
end:
```

Com isso, há um imediato aumento de desempenho simplesmente porque o processador não precisa carregar muitas instruções para fazer a mesma tarefa.

O QEMU é um exemplo de *software* que utiliza a técnica de emulação com recompilação dinâmica, e que pode ser usado para prover a virtualização.

Por fim, há uma diferença básica entre tradução dinâmica e a técnica de recompilação dinâmica (*dynarec*). A tradução dinâmica traduz um bloco de instruções do sistema convidado para instruções do sistema anfitrião anteriormente a execução do bloco e faz um *cache* dos blocos traduzidos para aumentar o desempenho. A técnica da recompilação dinâmica "descobre" qual algoritmo o sistema convidado implementa e substitui com uma versão otimizada no sistema anfitrião [KASICK et al].

3.7 As novas tecnologias de Virtualização da Intel e AMD

Tanto a Intel como a AMD vem investindo em tecnologia para permitir a virtualização em processadores de arquitetura x86. Elas introduziram arquiteturas similares, implementando um assistente em *hardware* (conhecido como *hardware assist*) para suportar virtualização de uma forma nativa, porém as tecnologias das duas fabricantes são incompatíveis entre si.

Com esta nova camada na CPU, o monitor de máquinas virtuais tem seu próprio nível privilegiado onde ele executa, e também não é mais necessário que seja feito o "desprivilegiamento" dos sistemas convidados, uma vez que os sistemas operacionais executam diretamente sobre o *hardware*.

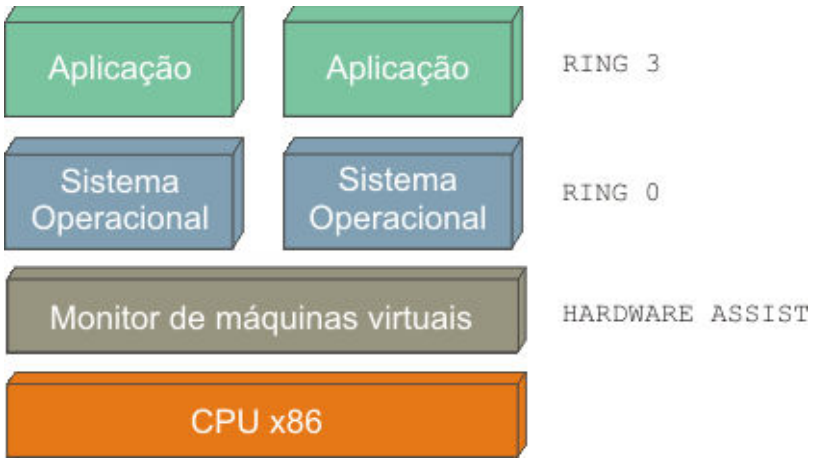


Figura 3.05 - Arquitetura x86 com o *hardware* assist da Intel e AMD

A Intel nomeou sua versão de seu *hardware assist* de "VT" ("*Virtualization Tecnology*") que anteriormente era chamado de "Vanderpool", enquanto a AMD

nomeou de “SVM” (“*Secure Virtual Machine*”) e mais recentemente mudou para “AMD-V” (“*AMD Virtualization”*), o codinome inicial era “Pacifica”.

A idéia é tornar suas CPU estritamente virtualizáveis, adicionando um modo hipervisor. Algumas instruções sensíveis ainda não geram *traps* no modo usuário, todavia isto não é uma grande preocupação quando o sistema convidado pode executar em modo supervisor.

Existem algumas diferenças entre as implementações da Intel e AMD, contudo os processadores dotados desta tecnologia basicamente possuem um conjunto de instruções extra chamado de Extensões de Máquina Virtual (*Virtual Machine Extensions - VMX*).

Por exemplo, no caso do Intel-VT, o VMX traz 10 novas instruções específicas para utilização de máquinas virtuais com o processador, são elas: VMPTRLD, VMPTRST, VMCLEAR, VMREAD, VMWRITE, VMCALL, VMLAUCH, VMRESUME, VMXOFF e VMXON.

Para entrar no modo de virtualização, o *software* deve executar a instrução VMXON e então chamar o monitor de máquinas virtuais. Feito isso, o monitor pode entrar em cada máquina virtual usando a instrução VMLAUNCH, e sair delas usando a instrução VMRESUME. Se o monitor quiser parar todas as máquinas virtuais e sair do modo de virtualização, ela executa a instrução VMXOFF.

O SVM disponibiliza a instrução VMRUN para executar uma máquina virtual, VMSAVE e VMLOAD são utilizadas para complementar as capacidades de guardar e restaurar estado da instrução VMRUN e dão acesso ao estado do processador. A instrução VMCALL permite que o sistema operacional se comunique diretamente com o monitor.

Em termos de gerenciamento de memória, existe uma diferença relevante entre a tecnologia da AMD e Intel. Nos processadores da AMD o gerenciamento de memória é feito por *hardware* enquanto que nos processadores da Intel ele é feito por *software*. Por esta razão, o desempenho da arquitetura SVM pode ser mais elevado que o da

arquitetura Intel-VT, embora esta eficiência ainda não tenha sido efetivamente comprovada [DUMIENSE e JESUS].

Outro avanço dessas tecnologias de virtualização é a inclusão de virtualização de E/S. Atualmente as soluções de virtualização por *software* não podem exclusivamente determinar que uma máquina virtual acesse diretamente um *hardware* físico (como uma placa de rede, por exemplo), tendo que simular estas operações com a utilização técnicas complexas [STEIL].

Desse modo, podemos concluir que a principal vantagem dessas tecnologias da Intel e AMD é uma melhora significativa de desempenho na virtualização de CPU, pois uma vez que há uma camada de virtualização nativa no *hardware*, não existe a necessidade de tradução binária.

CAPÍTULO IV

VIRTUALIZAÇÃO POR CAMADA DE ABSTRAÇÃO DO *HARDWARE*

4.1 Introdução

Neste conceito de virtualização existe uma camada de *software* entre o sistema operacional e o *hardware*, provendo a intercomunicação entre as partes. Esse *software* é o monitor de máquinas virtuais e geralmente pode suportar vários sistemas operacionais executando ao mesmo tempo com bom desempenho, característica que deve ser de fundamental importância para construção de *softwares* deste tipo.

A arquitetura dos monitores de máquinas virtuais que virtualizam por abstração do *hardware* pode ser de dois tipos principais, cujas diferenças já foram citadas no capítulo III, são elas: a virtualização total e a paravirtualização.

Esse tipo de virtualização está sendo muito difundido nos últimos anos e é atualmente a forma mais utilizada para virtualizar sistemas operacionais. Nesse capítulo são apresentados os principais *softwares* classificados nesse tipo de virtualização.

4.2 VMware

O *software* de virtualização mais difundido atualmente é o VMware. Lançado em 1999, foi a primeira solução de virtualização para computadores baseados na arquitetura x86. O VMware é um *software* proprietário que provê uma camada de virtualização que suporta vários sistemas operacionais sobre um *hardware*. A empresa desenvolvedora do VMware, a VMware Inc. é uma subsidiária da EMC Corporation e localiza-se em Palo Alto, Califórnia, Estados Unidos.

As principais versões do VMware são: VMware ESX Server, VMware Server, VMware Workstation e VMware Player.

4.2.1 VMware ESX Server

O VMware ESX Server é a versão comercial do produto VMware voltado para o uso em servidores de grande porte. Ele é uma máquina virtual do tipo I e possui um sistema operacional próprio e otimizado para gerenciar máquinas virtuais.

No VMware ESX Server, cada máquina virtual representa um sistema completo, com processador, memória, disco e BIOS, provendo um completo ambiente de execução, o que faz com que os sistemas operacionais convidados não precisem ser modificados.

Basicamente o sistema virtualiza quatro recursos chaves do servidor: CPU, memória, disco e rede [VMWARE 2006-A].

4.2.1.1 Virtualização de CPU

Como visto anteriormente, os processadores baseados na arquitetura x86 não possuem suporte nativo a virtualização, e por isso não conseguem capturar certas instruções de modo privilegiado originadas de um sistema operacional executando em uma máquina virtual. Desse modo, não podem contar totalmente com a técnica de captura e emulação dos sistemas tradicionais.

Para suprir esta dificuldade, o VMware usa adicionalmente a técnica de tradução binária. Com isso, o VMware examina todas as instruções antes de serem executadas, substituindo as instruções que não são causam *traps* por outras. Devido a isso, a técnica causa uma perda de desempenho, contudo há instruções que executam diretamente entre a máquina virtual e a CPU.

É importante salientar que o VMware dá a cada sistema operacional executando em sua máquina virtual, a sua própria CPU virtual, ou seja, os sistemas operacionais "acreditam" possuírem uma CPU dedicada. Cada CPU virtual possui seus próprios registradores e estruturas de controle.

4.2.1.2 Virtualização de memória

Como o ESX virtualiza a memória das máquinas virtuais por meio da tradução de endereços, o monitor de máquinas virtuais faz um mapeamento da página de memória do sistema operacional convidado para a página de memória física na subcamada do *hardware*. Cada máquina virtual tem sua própria página de memória em que o sistema operacional convidado "vê" iniciando do endereço 0.

O monitor intercepta instruções da máquina virtual que manipula estruturas de gerenciamento de memória do sistema operacional convidado, de modo que a unidade de gerenciamento da memória (*Memory Management Unit – MMU*) do processador não seja atualizada diretamente pelo sistema convidado. O ESX mapeia a página da máquina virtual em uma tabela de página sombra (*Shadow Page Table - SPT)* que é atualizado com o da máquina física.

Quando o sistema operacional convidado estabelece um mapeamento novo em sua tabela de página, o monitor detecta a modificação e atualiza a respectiva entrada na tabela de página sombra, que aponta para a localização real da página de memória no *hardware*. Quando a máquina virtual está executando, o *hardware* usa diretamente a tabela de página sombra para a tradução do endereço, o que permite que os acessos de memória normais na máquina virtual executem sem adicionar *overhead* de traduções de endereços, uma vez que as tabelas de página da sombra já estão definidas.

4.2.1.3 Virtualização de disco

O ESX Server implementa seu próprio sistema de arquivos chamado de VMFS. O VMFS é um sistema de arquivos distribuído que permite que múltiplos *hosts* acessem arquivos concorrentemente no mesmo volume VMFS. A principal vantagem é que o VMFS é otimizado para operações E/S com arquivos grandes como é o caso dos arquivos que contém as imagens de máquinas virtuais. Outro ponto importante é o

armazenamento em áreas de disco que podem ser compartilhadas para acesso entre os sistemas operacionais convidados executando em diferentes máquinas virtuais.

4.2.1.4 Virtualização de rede

No ESX Server é possível definir até quatro adaptadores de rede virtual. Cada adaptador tem seu próprio endereço MAC e endereço IP. As interfaces de rede virtual das múltiplas máquinas virtuais podem ser conectadas a um *switch* virtual. Cada *switch* pode ser configurado sem qualquer conexão ou a uma LAN física por meio dos adaptadores de rede da máquina anfitriã. Esses *switches* são chamados de "VMnets" e são abstrações que garantem conexões com velocidade entre as máquinas virtuais, o sistema anfitrião e a LAN física.

4.2.2 VMware Server

O VMware Server (que anteriormente chamado de VMware GSX Server) é a versão para uso em servidores de pequenos e médios portes. Tornou-se gratuito em 12 de junho de 2006 e disponibilizado para *download* no *site* oficial do fabricante.

O VMware Server é uma máquina virtual do tipo II, ou seja, é necessário que o *software* execute sobre um sistema operacional anfitrião que pode ser em sistemas operacionais baseados em Linux ou Windows (existe uma versão para cada um destas plataformas). O programa permite que sejam criadas diversas máquinas virtuais suportando alguns sistemas convidados de um modo otimizado, como por exemplo, algumas versões do Windows, Linux, Solaris e BSD (figura 4.01). Existe também um modo genérico para que se utilize outros sistemas operacionais sem suporte específico.

O VMware Server, assim como o VMware ESX, também suporta máquinas virtuais com uma ou duas CPU virtuais.

Ele pode compartilhar com os sistemas convidados, periféricos do *hardware* como: CD-ROM, placas de rede e portas USB.

Com ele existe a possibilidade de criar registros instantâneos (chamado de "snapshot") de uma máquina virtual num dado momento, no qual é possível fazer backup em um determinado estado, ou testar configurações em que se pode reverter.

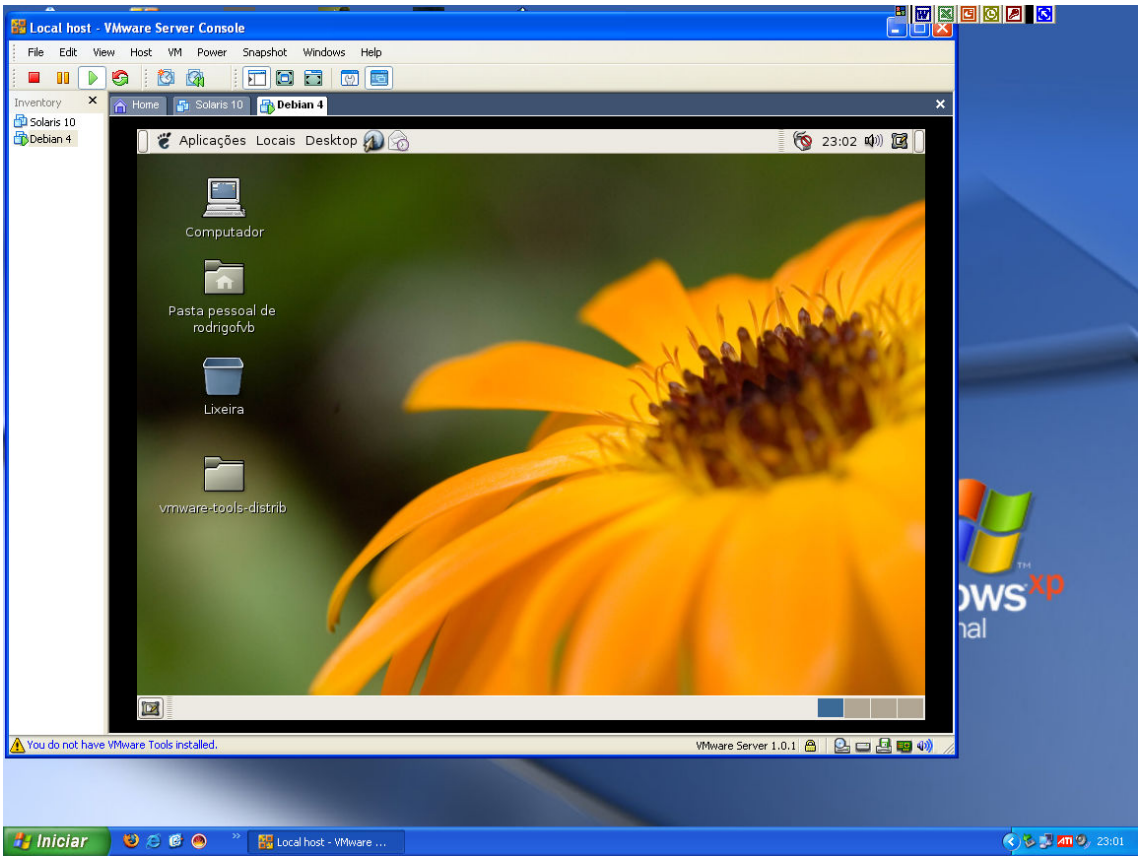


Figura 4.01 - VMware Server: Exemplo de uma máquina virtual Linux Debian executando sobre um sistema anfitrião Windows

No VMware Server o suporte a rede é feito através de VMnets (como no ESX Server), possuindo três modos:

- *Bridged*: a máquina virtual é vista como um outro computador na rede, com endereço IP podendo ser obtido via DHCP.
- NAT: a máquina virtual se conecta ao computador anfitrião, que por sua vez se conecta a rede.
- *Host-Only*: a máquina virtual apenas se conecta ao anfitrião.

Além disso, possui uma interface *web* para gerenciamento remoto. Para administração dos sistemas operacionais, o *software* usa uma versão modificada do VNC (figura 4.02).

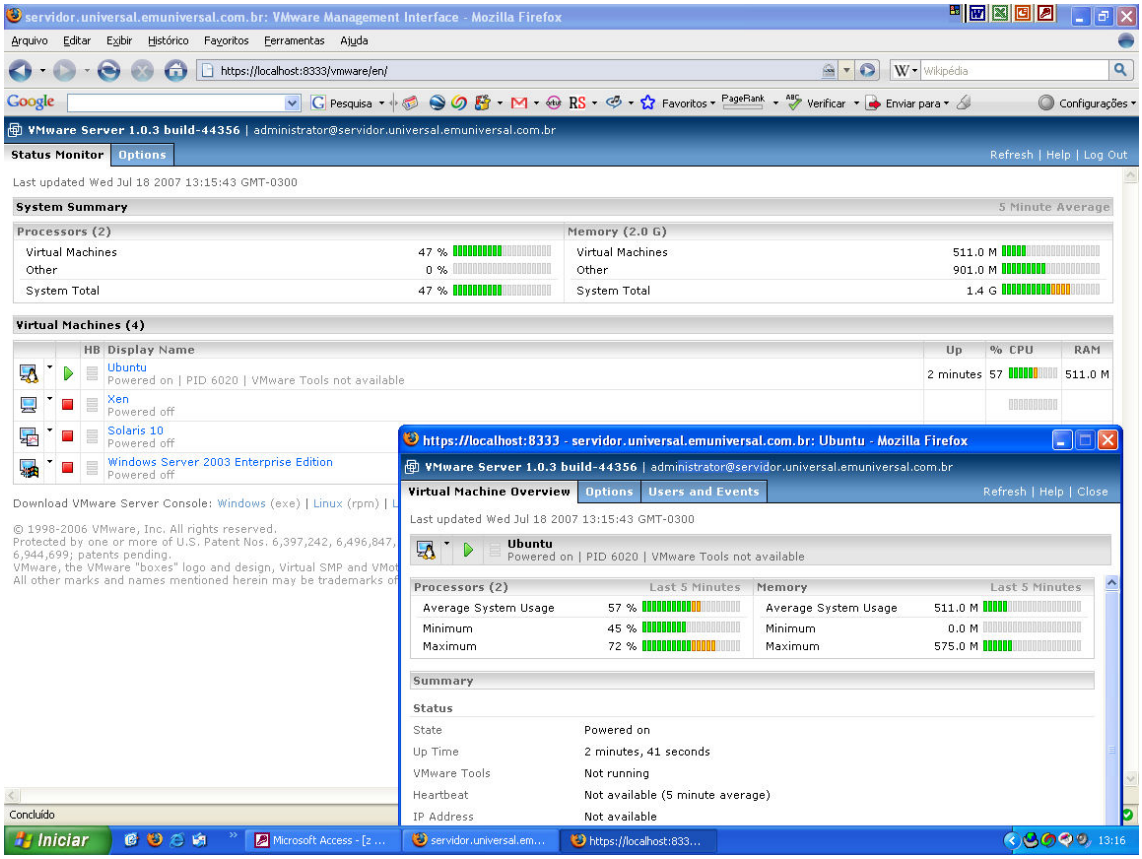


Figura 4.02 - VMware Server: Interface *web* de gerenciamento remoto.

4.2.3 VMware Workstation

Esta é a versão comercial do VMware que é utilizada em estações de trabalho. Possui basicamente os mesmos recursos do VMware Server inclusive com a possibilidade de criar máquinas virtuais.

O VMware Workstation destaca-se pela facilidade de uso proporcionada por seus assistentes que guiam o usuário no processo de criação de máquinas virtuais. Ele também possui um assistente que ajuda a montar clones de máquinas virtuais. Também é possível criar grupos de máquinas virtuais, de uma só vez, e colocá-las em redes.

Com o VMware Workstation é possível criar máquinas virtuais em dispositivos externo como um disco rígido ou um *pen-drive*, através de um produto adicional chamado ACE *Option Pack.*

4.2.4 VMware Player

Esta é a versão mais simples do produto e que também é disponibilizada gratuitamente para *download*. O VMware Player é indicado para aplicações leves e não pode criar máquinas virtuais, porém executa as máquinas virtuais criadas por outras versões mais completas.

4.2.5 VMWare Infrastructure

O VMware Infrastructure não é uma versão, na verdade ele é uma suíte de aplicativos de virtualização que otimiza o processo de implantação e gerenciamento de máquinas virtuais nas empresas. Ele visa oferecer, em uma solução integrada, ganhos em: virtualização compreensiva, gerenciamento, otimização de recursos e disponibilidade de aplicações. A suíte é composta de um conjunto de aplicativos cujo principal é o VMware ESX Server. São eles:

- VMware ESX Server: já visto anteriormente, o ESX é um *software* que provê uma camada de virtualização, no qual abstrai processador, memória, disco e recursos de rede. O ESX possui ainda, um sistema operacional próprio, o que visa aumentar o desempenho das máquinas virtuais.
- Virtual Symmetric Multi Processing (SMP): possibilita que uma simples máquina virtual use múltiplos processadores simultaneamente.
- Virtual Center Management Server: um ponto central para configurar e gerenciar infra-estrutura de TI virtualizada.
- Virtual Infrastructure Client: uma interface que permite administradores e usuários se conectarem remotamente ao *Virtual Center Management Server* ou instalações individuais do ESX.
- Virtual Infrastructure Web Access: Uma interface *web* para gerenciamento e acesso remoto.
- VMotion: habilita uma migração em tempo de execução de uma máquina virtual de um servidor para outro com a menor queda possível no nível de serviço.

- High Availability (HA): mantém a alta disponibilidade das máquinas virtuais. Com ele, em caso de falha de um servidor as máquinas virtuais afetadas são automaticamente reiniciadas em outros servidores de produção que possuam condições para abrigá-las.
- Distributed Resource Scheduler (DRS): Distribui de forma inteligente recursos de *hardware* para as máquinas virtuais.
- Consolidated Backup: um agente centralizado para *backup* de máquinas virtuais. Ele simplifica a administração e reduz a carga nas instalações do ESX Server. [VMWARE 2006-B].

Dentro da suíte VMware Infrastructure, o *Virtual Center* é um aplicativo que é uma central administrativa do ESX Server. Ele visa facilitar e agilizar a criação de máquinas virtuais pois possui uma série de *templates* (máquinas virtuais criadas previamente) de servidores prontos. Com o *Virtual Center* é possível criar máquinas virtuais através de um modelo (com um servidor de banco de dados já instalado, por exemplo) e deixar o *software* implantar o novo sistema em servidores físicos, o que torna a tarefa mais rápida do que executar uma instalação e configuração de um novo servidor físico na rede. A figura 4.03 ilustra um exemplo de funcionamento do VMware Virtual Center.

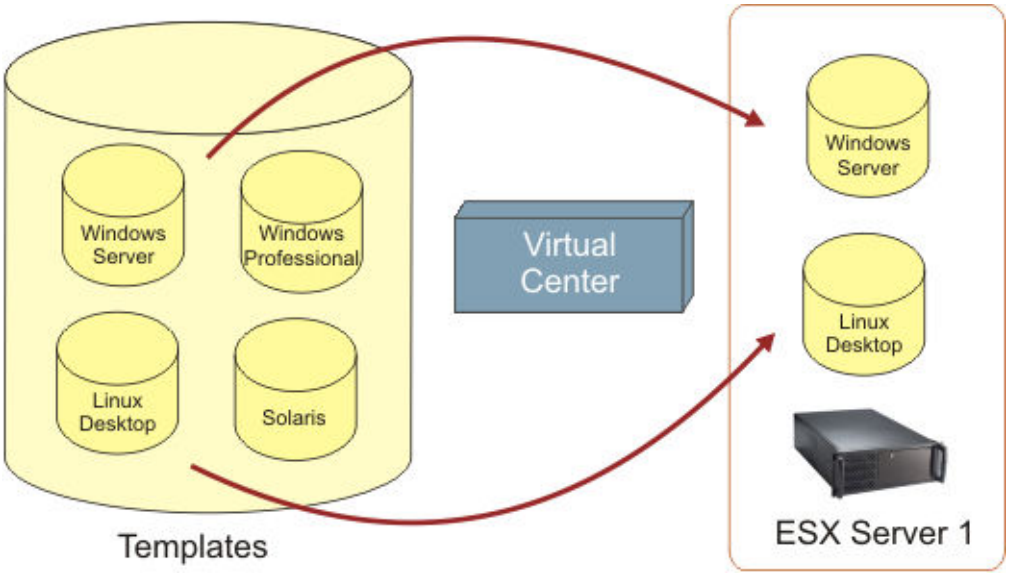


Figura 4.03 - Criação de máquinas virtuais com o VMware Virtual Center

4.2.5.1 Alta Disponibilidade

O *High Availability* é uma aplicação que permite a migração de máquinas virtuais em caso de falha do servidor físico, para um outro que esteja operacional no momento.

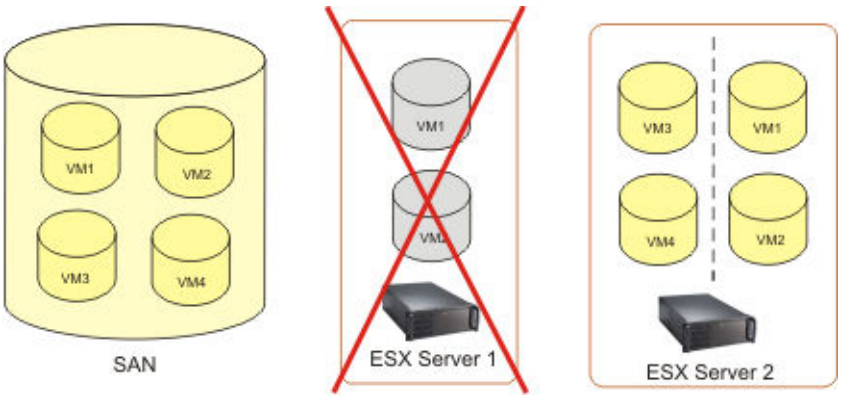


Figura 4.04 - Infra-estrutura de máquinas virtuais com alta disponibilidade

A figura 4.04 ilustra um exemplo de uma infra-estrutura em que dois servidores compartilham uma rede de armazenamento de dados (SAN[3]). Nesta configuração, os arquivos físicos das máquinas virtuais estão armazenados na SAN e cada CPU tem acesso simultâneo às informações. No caso de pane de um dos servidores, as máquinas virtuais são transferidas de um servidor a outro. O VMotion, que é outra aplicação incluída na suíte, visa executar estas transferências com o menor *downtime* possível.

O VMotion também ajuda no processo de transferir máquinas virtuais de um servidor para outro (figura 4.05) com o intuito de aliviar a carga deste primeiro em momentos de picos de utilização. Ele também auxilia no processo de mover máquinas virtuais para servidores novos.

Também é possível agendar trocas de máquinas virtuais, ou configurar as trocas para que sejam feitas em função dos picos de utilização, com a aplicação chamada *Distributed Resource Scheduling* (DRS). Sua principal função é eliminar a ociosidade

[3] SAN (*Storage Area Network*) é uma rede projetada para agrupar vários dispositivos de armazenamento de informações.

das máquinas, redistribuindo os recursos de forma a manter a disponibilidade e o máximo desempenho do parque.

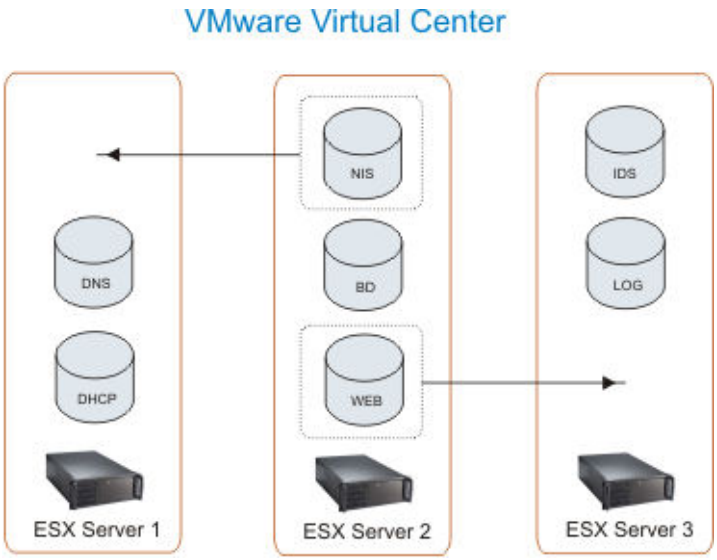


Figura 4.05 - Migração de máquinas virtuais entre servidores

4.2.6 Teste do VMWare

Em meu teste foi utilizada a versão VMware Server (que possui licenciamento gratuito), o qual instalei sobre um sistema anfitrião Windows XP. Em seguida foram criadas duas máquinas virtuais contendo os seguintes sistemas operacionais convidados: Debian 4.0 e Ubuntu 6.06 Server.

Para criação dessas máquinas virtuais foram utilizados dois discos virtuais novos para abrigar cada instalação e o escolhido modo de conexão a rede *bridged* para ambas.

Ao final do processo pude concluir que:

- Ambos os sistemas foram instalados sem apresentar problemas e funcionaram normalmente.
- O desempenho foi bom, mesmo com as duas máquinas virtuais executando simultaneamente no sistema anfitrião.
- Todas as máquinas acessaram normalmente a rede, sendo que para cada uma foi configurado um endereço IP distinto, os quais responderam aos comandos `ping`

originados de outra máquina virtual e do sistema anfitrião, conforme pode ser visto na figura 4.06.

- Ficou evidente que o VMware Server deve ser utilizado preferencialmente em redes de pequeno porte ou em estações de trabalho, devido a sua simplicidade e carência de grandes recursos.
- O VMware mesmo em sua versão limitada é um ótimo *software* de virtualização pois alia bom desempenho, compatibilidade com os diversos sistemas operacionais e facilidade de uso.

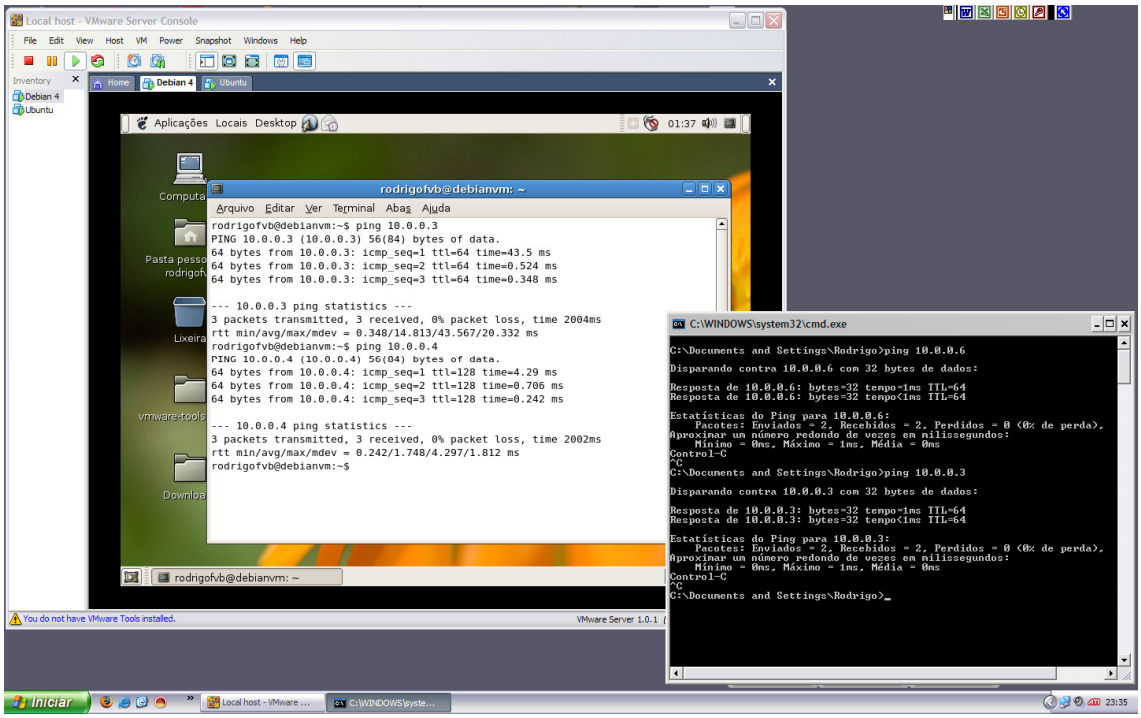


Figura 4.06 - Execução de comandos `ping` entre as máquinas virtuais e o sistema anfitrião

4.3 Xen

O Xen é uma plataforma de virtualização do tipo I para a arquitetura x86. O projeto Xen é de código aberto e baseado na *General Public License* (GPL). Foi originalmente parte chave de um projeto de pesquisa na Universidade de Cambridge chamado XenoServer, cujo objetivo era "prover uma infra-estrutura pública para computação distribuída" [CLARK]. Ian Pratt, que foi líder do projeto, fundou a empresa XenSource Inc. que atualmente suporta o desenvolvimento do projeto *open source* e também possui versões comerciais do *software*.

O Xen utiliza o modelo de paravirtualização para prover máquinas virtuais. Como já foi visto, a paravirtualização é o processo que simplifica a interface exportada para *hardware* de uma forma que elimina certas características que são difíceis de virtualizar. Um exemplo dessas características são as instruções sensíveis que possuem um comportamento diferente dependendo da forma que está executando nos modos usuário ou *kernel*. Um significante *overhead* é gerado uma vez que as instruções devem ser interceptadas e interpretadas pela camada de virtualização. O Xen utiliza a paravirtualização para reduzir esse *overhead* [YOUSEFF et al]. A paravirtualização permite que máquinas virtuais específicas comuniquem-se diretamente com o *hardware*. Ao invés de todas as máquinas virtuais se comunicarem com o sistema anfitrião, é a máquina privilegiada que gerencia a interação e recebe as chamadas passadas pelos outros sistemas virtuais.

Ele possui acesso privilegiado ao *hardware* e os sistemas convidados utilizam esse acesso privilegiado como uma espécie de ponte para acessar o *hardware*. A memória é separada em blocos pelo monitor de máquinas virtuais, e os sistemas convidados podem utilizar esses blocos da forma que lhes convier, tornando o acesso à memória direto e mais eficiente. Esta separação e isolamento também ocorrem em dispositivos como disco, o que proporciona um acesso mais rápido a estes. O monitor de máquinas virtuais só interfere quando ocorre um acesso fora do bloco [LAUREANO 2006].

É necessário que os sistemas operacionais sejam modificados para suportar esta virtualização, contudo não é necessário que as aplicações que executem sobre esses sistemas sejam reescritas. Algumas distribuições Linux como o Red Hat e o Debian, já possuem suporte nativo ao Xen.

É necessário frisar que o Xen suporta (através de pacotes adicionais) que sistemas operacionais executem sem modificações, desde que o *hardware* possua processadores com a tecnologia de *hardware assist* (já comentado no capítulo III), como o Intel VT da Intel e AMD-V da AMD [CLARK].

Na paravirtualização em interfaces x86, existem quatro fatores amplos: Os gerenciamentos de memória, da CPU, dos dispositivos E/S e de redes [BARHAM et al].

4.3.1 Gerenciamento de Memória

A virtualização de memória representa a maior dificuldade em paravirtualizar uma arquitetura, isto por causa de mecanismos requeridos no monitor e das modificações requeridas nos sistemas convidados.

Em cada máquina virtual criada no Xen (denominado “domínio” ou “dom”), uma tabela de página de memória é alocada pelo monitor, que a partir daí usa um mecanismo chamado *hypercall* para realizar as atualizações nessas tabelas. O monitor (que executa no domínio 0 ou dom0) permite que os sistemas operacionais convidados acessem tabelas de páginas diretamente, porém em modo somente para leitura.

4.3.2 Gerenciamento da CPU

Como visto no capítulo III, a arquitetura x86 implementa camadas de privilégio genericamente descritas como anéis (*rings*), enumeradas de 0 (mais privilegiada) até 3 (menos privilegiada).

O Xen utiliza a técnica de desprivilegiamento, no qual o monitor que executa no *ring* 0 desloca a execução do sistema operacional convidado para o *ring* 1 ou 2 para que ele possa manter um controle das instruções privilegiadas. As aplicações de usuário continuam executando no *ring* 3.

As instruções privilegiadas são paravirtualizadas de uma forma que requer que elas sejam validadas e executadas pelo monitor. Qualquer sistema operacional convidado que tente executar diretamente uma instrução privilegiada causa uma falha, uma vez que somente o Xen executa em nível privilegiado suficiente.

4.3.3 Dispositivos de E/S

O gerenciamento de dispositivos E/S é feito basicamente emulando dispositivos de *hardware* existentes como é tipicamente feito na virtualização total. O Xen implementa

um conjunto de abstrações simples para os dispositivos, que permite projetar uma interface que seja eficiente e satisfatória quanto à proteção e isolamento.

Somente o domínio 0 (dom0) tem acesso direto sem checagem aos discos físicos. Todos os outros domínios acessam o disco através de uma abstração chamada VBD (*Virtual Block Devices*), que são criadas e configuradas pelo monitor (que executa dentro do domínio 0).

Um VBD compreende uma lista de extensões com associações e informações de controle de acesso, e aparece para os sistemas operacionais como um disco SCSI.

O "mecanismo" consiste em manter uma tabela de tradução com o monitor para cada VBD. Ao receber uma requisição de disco, o Xen inspeciona o identificador VBD, checa as permissões, e define o endereço correspondente do setor e o dispositivo físico do armazenamento.

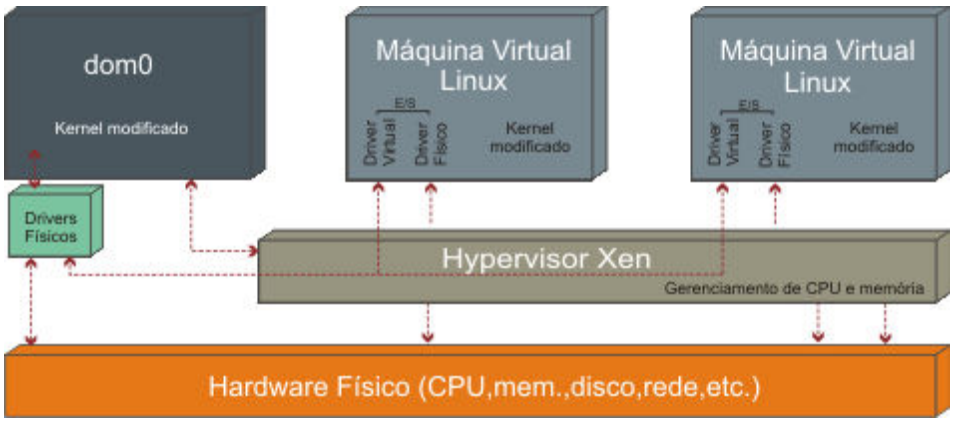


Figura 4.07 - Xen: Dom0 acessa o *hardware*, os outros domínios são restritos a disco e rede virtuais

4.3.4 Dispositivos de Rede

O Xen provê uma abstração de um roteador/*firewall* virtual (*Virtual Firewall Router* - VFR) onde cada domínio tem uma ou mais VIF (*Virtual network InterFace*), que seriam as placas de rede (virtuais) de cada sistema operacional hospedado.

Para transmitir um pacote, o sistema operacional convidado simplesmente enfileira um descritor de *buffer* no anel de transmissão. O Xen copia o descritor, examina o cabeçalho do pacote e executa quaisquer regras de filtros de pacotes que haja.

Quando o pacote é recebido, o Xen imediatamente checa as definições de regras de recepção para determinar a VIF de destino, e troca o *buffer* do pacote por um quadro no anel de recepção. Se nenhum quadro estiver disponível, o pacote é descartado [PRZYBYSZ e LUIZ].

4.3.5 Migração de domínios

A migração que é usada para transferir um domínio entre *hosts* físicos é um dos recursos mais úteis no Xen, pois dá a possibilidade de agendar uma manutenção planejada do *hardware*, ou transferir a carga de uma máquina sobrecarregada, ou apenas mudar um serviço específico para uma nova máquina.

A migração no Xen pode ser feita de duas formas: regular e *live*. O primeiro move uma máquina virtual de um *host* para outro fazendo uma pausa nela, copiando o conteúdo na memória e reiniciando no destino. O último executa a mesma funcionalidade lógica, contudo sem a necessidade de pausar o domínio. Com a migração *live* o domínio continua sua atividade, o que da perspectiva do usuário a migração é imperceptível.

4.3.6 Teste do Xen

Implementei o Xen da seguinte forma (figura 4.08): um sistema anfitrião (dom0) Debian, e duas máquinas virtuais OpenSuse Linux (dom1 e dom2). Vemos na figura uma janela de *status* (acima), com os dados de consumo (processamento, memória, disco etc) de ambas as máquinas virtuais e também do dom0. Neste caso o Xen foi executado sobre um *kernel* do Debian, contudo ao contrário do exemplo, não é necessário executar o Xen com uma interface gráfica uma vez que ele executa normalmente em modo texto, consumindo menos recursos.

Neste teste, os três sistemas executaram com excelente desempenho. A configuração foi relativamente fácil devido a presença de um assistente de instalação e configuração. O sistema apresentou boa estabilidade mesmo com altos picos de utilização de CPU e memória, os quais puderam ser devidamente acompanhados pela janela de *status* dos sistemas virtuais.

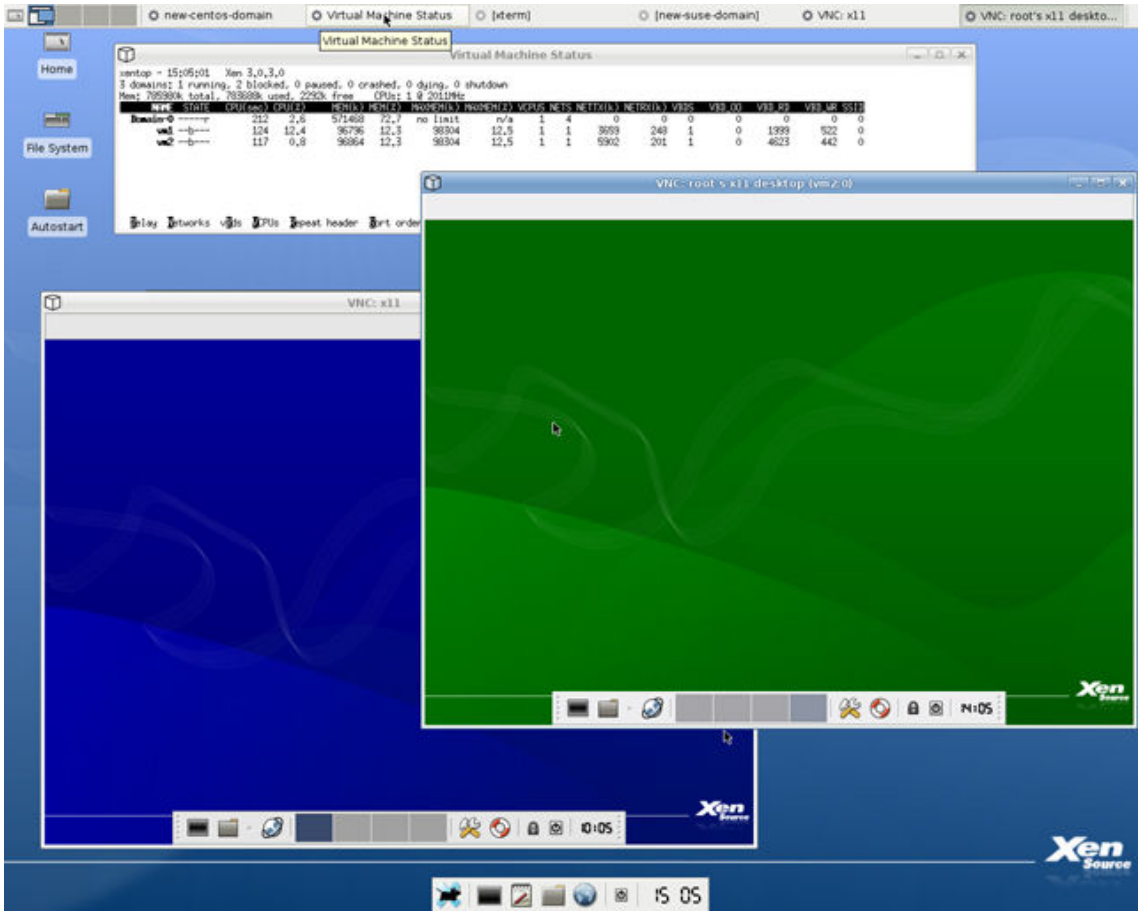


Figura 4.08 - Ambiente do Xen com duas máquinas virtuais

4.4 Microsoft Virtual PC e Microsoft Virtual Server

O Virtual PC é um *software* da Microsoft lançado em dezembro de 2003 para virtualizar sistemas operacionais Windows ou emulá-los em sistemas baseados no PowerPC (Macintosh). O programa foi originalmente escrito em 1997 pela empresa Connectix que foi adquirida pela Microsoft posteriormente.

Tecnicamente o Virtual PC emula um processador Intel Pentium de 32 bits com um chipset Intel 440BX, um adaptador gráfico padrão SVGA VESA (com 16 MBytes de VRAM, em suas últimas versões), um BIOS da American Megatrends (AMI), um adaptador de som Creative Labs Sound Blaster 16 PnP, e uma placa de rede DEC 21140 (Julho 2007).

A versão Macintosh do Virtual PC usa a recompilação dinâmica para traduzir o código x86 usado por um PC padrão para seu equivalente no PowerPC. A versão Windows do Virtual PC também usa recompilação dinâmica, mas apenas para traduzir o modo supervisor (*kernel mode*) em código de modo usuário (*user mode*) enquanto o modo usuário original e o código do modo virtual executa nativamente.

Para competir com o VMWare ESX Server, a Microsoft lançou uma versão do Virtual PC para servidores Windows denominado Microsoft Virtual Server. Em janeiro de 2006, a Microsoft reduziu de forma significativa os preços das versões do Virtual Server. A VMWare por sua vez lançou o VMware Server (anteriormente VMware GSX Server) a custo zero. Durante o LinuxWorld Trade Show em abril de 2006, a Microsoft anunciou uma nova versão do Virtual Server (Virtual Server 2005 R2 Enterprise Edition), também com custo zero e suporte nativo à plataforma x64 [HIGASHIYAMA]. Atualmente a Microsoft disponibiliza o Virtual PC e o Virtual Server para *download* gratuito em seu *site* na Internet.

As principais características do Virtual Server são [MICROSOFT]:

- Funciona somente em servidores Windows, suportando qualquer versão de Windows como máquina virtual.
- Suporte para conectividade permitindo *cluster* de todas as máquinas virtuais executando sobre um *host*.
- Suporta a tecnologia 64 bit.
- Melhorias no *hyper-threading*.
- Integração com o *Active Directory*.
- Possibilidade de migração de máquinas virtuais com ferramentas especiais.
- *Virtual Hard Disks* (VHD): oferece flexibilidade ao encapsular máquinas virtuais em discos virtuais,

4.4.1 Discos virtuais (*Virtual Hard Disk*)

As formas como podem ser armazenadas as máquinas virtuais no Microsoft Virtual Server são uns dos recursos interessantes do *software*, pois chama atenção pela flexibilidade.

No Virtual Server podemos criar os discos virtuais previamente (que no sistema Windows anfitrião são arquivos que ficam com a extensão ".vhd"), mantendo uma lista de discos virtuais disponíveis que podem ser utilizados a qualquer momento, ou diretamente no momento de criação de uma máquina virtual. As opções de alocação de disco são:

- Alocação dinâmica (*Dynamically expandig*): O tamanho arquivo do disco virtual (VHD) aumenta conforme os dados são gravados nele. O tamanho inicial é tipicamente 100 *kilobytes*, mas conforme os dados são adicionados o disco irá aumentar até que alcance o limite especificado no momento da criação. É o padrão para criação de máquinas virtuais.
- Tamanho fixo (*Fixed-size*): O tamanho do arquivo VHD é um tamanho fixo especificado no momento de criação do disco virtual. Por exemplo: se na criação do disco virtual for definido o tamanho fixo de 5 *gigabytes*, arquivo VHD imediatamente possuirá o tamanho de 5 *gigabytes.*
- Diferenciado (*Differencing*): o modo diferenciado permite possuir múltiplas configurações de sistemas operacionais que são baseadas em uma única instalação de sistema operacional. Um disco virtual diferenciado é um disco virtual associado com outro disco virtual numa relação pai e filho. Nesta analogia, o disco virtual é o filho e o disco virtual associado é o pai.
- Vinculado (*Linked*): Este modo é um vínculo entre um disco virtual e um disco físico. O sistema convidado acessa diretamente os arquivos armazenados no disco físico.

4.4.2 Teste do Microsoft Virtual Server

Nesse teste fiz uma instalação de uma versão do Microsoft Virtual Server que obtive através dá pagina da Microsoft na Internet. O sistema instalou sem problemas, integrando-se ao sistema operacional Windows Server 2003 anfitrião.

A figura 4.09 mostra a interface *web* de administração e criação de máquinas virtuais do virtual Server, o qual executa diretamente sobre o servidor *web* do próprio Windows Servidor (IIS).

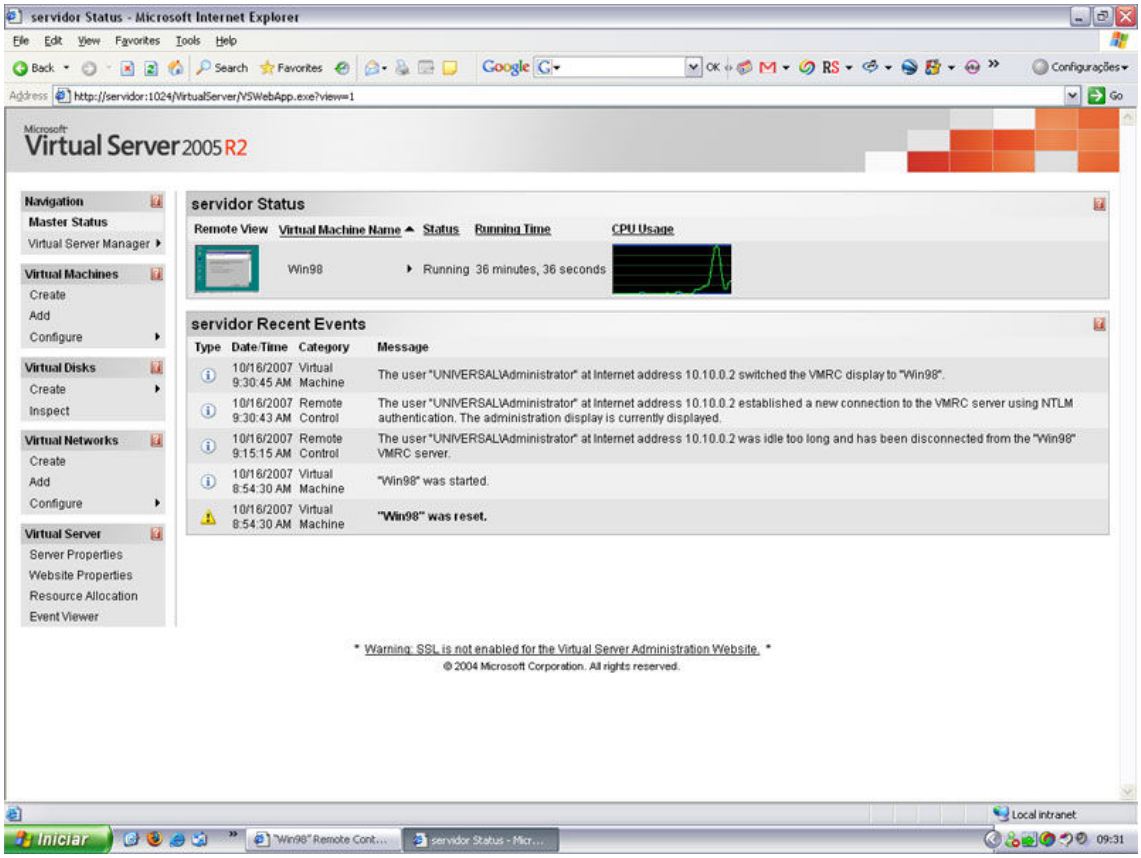


Figura 4.09 - Interface *web* de administração do Virtual Server

Após a instalação do *software*, fiz uma instalação do Windows 98 como sistema convidado, o que também ocorreu sem problemas. A figura 4.10 mostra o funcionamento do teste.

Para esta instalação foi utilizado o modelo de crescimento dinâmico para o armazenamento da máquina virtual, o que se mostrou muito interessante uma vez que o tamanho do arquivo foi alocado dinamicamente pelo monitor, conforme realmente necessário.

O Microsoft Virtual Server é, sem dúvida, uma boa opção para virtualização de sistemas totalmente baseados em Windows, uma vez que possui suporte e integração a servidores Windows, é gratuito, possui boa flexibilidade, além de ser da mesma empresa fabricante dos sistemas operacionais, o que em teoria, agrega melhor compatibilidade e confiabilidade ao produto.

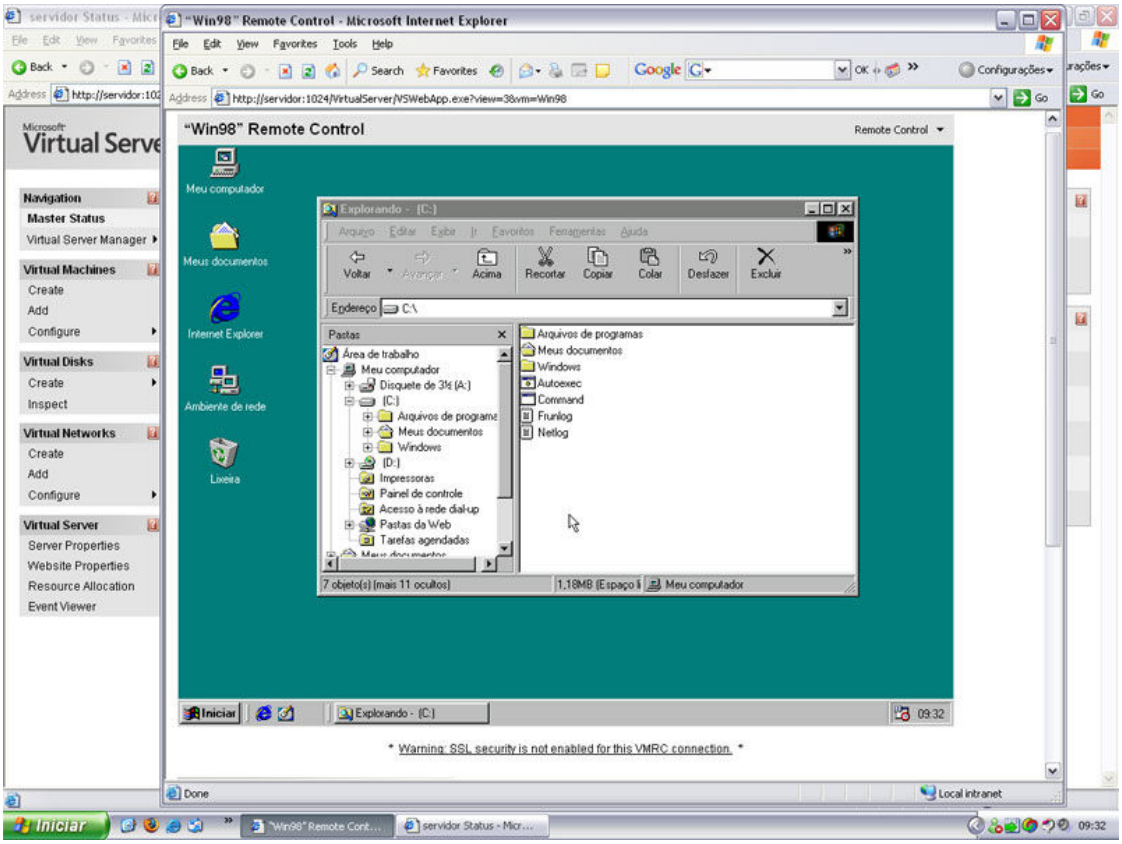


Figura 4.10 - Sistema Operacional convidado Windows 98 executando no Virtual Server

4.5 VirtualBox

O VirtualBox é um *software* de virtualização para arquitetura x86 desenvolvido pela empresa Innotek GmbH, com sede na Alemanha. Desde janeiro de 2007 possui uma versão que é *open source* de licença GNU GPL.

O *software* é uma máquina virtual do tipo II e executa como um processo de sistema operacional *host* que pode ser Linux, Windows 32 ou 64 bits, ou Mac OS X. Atualmente suporta sistemas convidados como DOS, FreeBSD, Linux, OpenBSD, NetBSD, Solaris, Netware, OS/2 Warp e Windows.

Ele utiliza a técnica da virtualização total, emulando componentes chaves do *hardware*. Com isso, não há necessidade de que os sistemas operacionais convidados sejam modificados para que executem em uma máquina virtual.

O VirtualBox tenta executar uma porção do código dos sistemas virtuais diretamente no processador. Para que sejam executadas as instruções privilegiadas, ele tenta mover o sistema operacional para o nível de *ring* 1, ao invés do *ring* 0. Como já visto, o nível de *ring* 1 geralmente não é utilizado na arquitetura x86. Caso haja problemas no processo, o VirtualBox também utiliza a técnica de recompilação dinâmica. O recompilador do VirtualBox é baseado no *open-source* QEMU (que será visto mais adiante). Adicionalmente, o VirtualBox automaticamente "desmonta" e, na maioria dos casos, "corrige" o código dos sistema convidados a fim de prevenir futuras recompilações. Em razão disso o código executa nativamente na maior parte do tempo, numa tentativa de aumentar seu desempenho [VIRTUALBOX].

No *software*, os discos são emulados num recipiente especial chamado *Virtual Disk Image* (arquivos VDI), o qual até o momento é incompatível com formatos usados por outras soluções. O VirtualBox possui uma funcionalidade que pode conectar dispositivos iSCSI e usá-los como discos virtuais.

O VirtualBox virtualiza os adaptadores gráficos como no padrão VESA e cuja memória pode ser ajustada. Em sistemas convidados Linux e Windows, pode ser instalado *drivers* gráficos especiais para melhorar o desempenho.

Os adaptadores de rede são virtualizados como adaptadores AMD PCNet, e placas de som como dispositivos Intel ICH AC'97. Dispositivos USB também são emulados.

Outras características do VirtualBox:

- Permite virtualização recursiva (uma instância do VirtualBox pode ser executada em sistema convidado).
- Permite controle total através de linha de comando.
- Permite *logon* automático em máquinas virtuais Windows.
- Inclui um servidor *Microsoft Remote Desktop Protocol* (RDP) para administração de máquinas virtuais.
- Inclui suporte total a Intel VT e suporte experimental ao AMD-V.

4.5.1 Teste do VirtualBox

Em meu teste, o VirtualBox mostrou-se bem simples. Possui algumas das opções de configuração de máquina virtual presentes no VMware Server. Contudo ainda não possui recursos como: maior flexibilidade na configuração da rede virtual, opções de inicialização automática do sistema virtual junto com o sistema anfitrião e suspensão e reativação da máquina virtual sem a necessidade de desligá-la.

O VirtualBox foi instalado em PC com Windows XP como sistema anfitrião, e foi criada uma máquina virtual, o qual recebeu uma versão do sistema operacional Kurumin Linux que é baseado no Debian (figura 4.11). Esse sistema operacional funcionou perfeitamente, entretanto quando tentei criar outra máquina virtual para abrigar o sistema operacional FreeBSD, este não foi possível instalar, acusando erro em todas as tentativas. A documentação do programa afirma que há suporte ao sistema operacional FreeBSD.

Nesse teste pude concluir que o VirtualBox ainda é um sistema bem simples se comparado ao VMware, e por isso ainda não é recomendado utilizá-lo em ambientes de servidores. Todavia, ele é uma boa opção para um usuário que necessite utilizar sistemas operacionais virtuais, uma vez que possui bom desempenho, que foi aferido no teste.

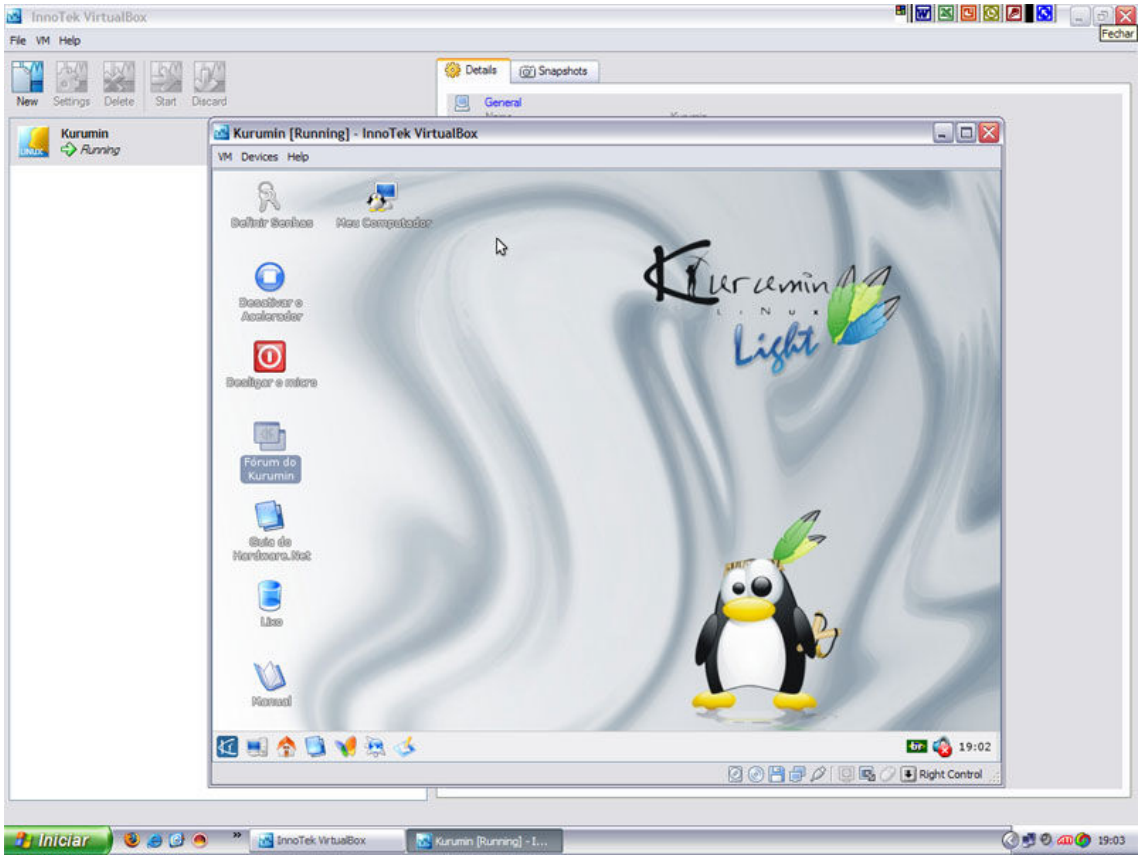


Figura 4.11 - Interface de gerenciamento de máquinas virtuais do VirtualBox com o Kurumin Linux executando como sistema convidado

4.6 User-Mode Linux

O User-mode Linux (UML) é um espaço de usuário (*userspace*) portado do *kernel* do Linux. Ele foi idealizado por Jeff Dike em 1999 e é uma máquina virtual de tipo II, já que cada máquina virtual executa como um processo do sistema Linux anfitrião. A máquina virtual do User-mode Linux é capaz de executar o mesmo conjunto de aplicações e serviços que executam nos sistema anfitrião.

Características do User-mode Linux [YEHUDA]:

- Executa sistemas operacionais sem necessidade de modificações.
- Pode executar diferentes *kernels* ou distribuições Linux (não há necessidade que o sistema convidado seja igual ao sistema anfitrião).
- Máquina virtual segura.
- Bom desempenho dependendo da carga de trabalho.
- Acessa o sistema de arquivos do sistema anfitrião (via hostfs).

- Acesso total à rede.
- Suporta SMP (*Symmetric Multiprocessing*).

O monitor de máquinas virtuais do User-mode Linux executa como um processo que controla as máquinas virtuais, e por sua vez, cada máquina virtual executa como um processo do sistema anfitrião. O *kernel* do sistema convidado não se comunica diretamente com o *hardware*. Ao invés disso, essa comunicação é passada para o *kernel* do sistema anfitrião, o qual verdadeiramente acessa o *hardware*.

Uma vez que tanto o sistema virtual como o sistema anfitrião são sistemas Linux (com estruturas quase idênticas), a comunicação é feita de forma muito eficiente, necessitando de pouca abstração e tradução.

4.6.1 Chamadas de sistema

Como o Linux não fornece nenhum mecanismo de distinção entre o modo usuário e o modo supervisor (*kernel*), o UML usa a chamada de sistema de rastreamento *ptrace* para prover seu próprio mecanismo. O UML tem uma *thread* especial cujo principal trabalho é controlar a execução de todos os outros processos. Quando um processo está em modo usuário, suas chamadas de sistema estão sendo interceptadas pela *thread* de rastreamento. Quando está no modo *kernel*, ele não está sendo rastreado. Desta forma o UML faz a distinção entre o modo usuário e o modo supervisor.

Quando este mecanismo intercepta uma chamada de sistema de um processo e troca entre modo usuário e modo supervisor, a virtualização das chamadas de sistema acontece diretamente, ou seja, a *thread* de rastreamento anula a chamada de sistema no *kernel* anfitrião. Após isto, ela salva o estado do processo e impõe um novo estado, forçando que o processo inicie a execução da chamada de sistema. Ao final da execução, um sinal é enviado. Logo após isto, o processo armazena os valores de retorno e requisita a *thread* de rastreamento que o retorne para o modo usuário [DIKE].

Como foi visto, é a *thread* de rastreamento que redireciona todas as chamadas de sistema para o *kernel* virtual.

4.6.2 Sistema de arquivos

O UML tem um sistema de arquivos (chamado “hostfs”) que provê acesso direto ao sistema de arquivos anfitrião. Isto é possível, pela implementação da interface VFS (*Virtual File System*), que traduz todas as chamadas equivalentes no sistema anfitrião.

O hostfs também pode ser um sistema de arquivos raiz, sendo registrado como um sistema de arquivos virtual [DIKE].

4.6.3 Desempenho

O UML não precisa emular um *hardware* específico, ao invés disso, as instruções são passadas com eficácia entre o *kernel* virtual e o *kernel* anfitrião. Com isso, o UML pode executar código nativo, e na pior hipótese, com uma perda de desempenho de apenas 20% (vinte por cento) em comparação com o mesmo código sendo executado no sistema anfitrião [LINUX].

4.6.4 Teste do User-Mode Linux

O UML está presente nativamente nos *kernel* Linux da série 2.6. Nos sistemas cuja versão do *kernel* seja inferior, deve ser adquirido os módulos do UML para compilar com o *kernel.* Pode-se também obter uma versão do *kernel* inferior ao 2.6 já compilado, que é disponibilizado no *site* oficial do projeto.

Em meu teste utilizei o sistema operacional Linux Debian 4.0, o qual já possui suporte ao UML. A seguir, baixei um sistema de arquivo do Fedora já pronto e disponível no *site* oficial do projeto. Para a inicialização da máquina virtual o comando dado foi:

```
# linux ubda=FedoraCore5-x86-root_fs mem=128M
```

Desse modo uma máquina virtual contendo o Fedora foi iniciada sobre o sistema anfitrião Debian, conforme visto na figura 4.12. A figura 4.13 mostra as pastas do

sistema de arquivos do Fedora totalmente isoladas do sistema anfitrião na máquina virtual.

Nesse teste, o UML mostrou que é uma boa opção para virtualização em sistemas Linux com *kernel* igual ou mesmo outro *kernel* Linux, uma vez que já é suportado nativamente pelas distribuições atuais e pela sua facilidade de uso.

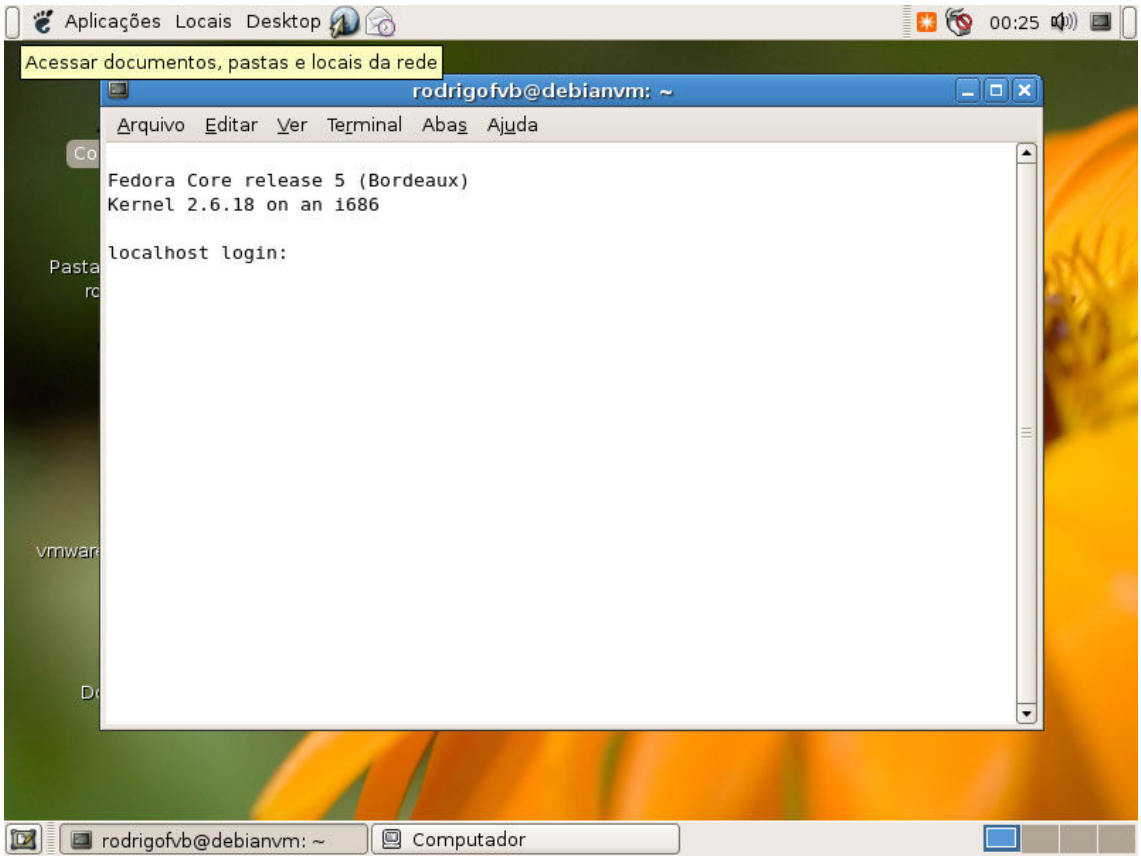


Figura 4.12 - Tela de *login* da máquina virtual Fedora sobre o sistema anfitrião Debian

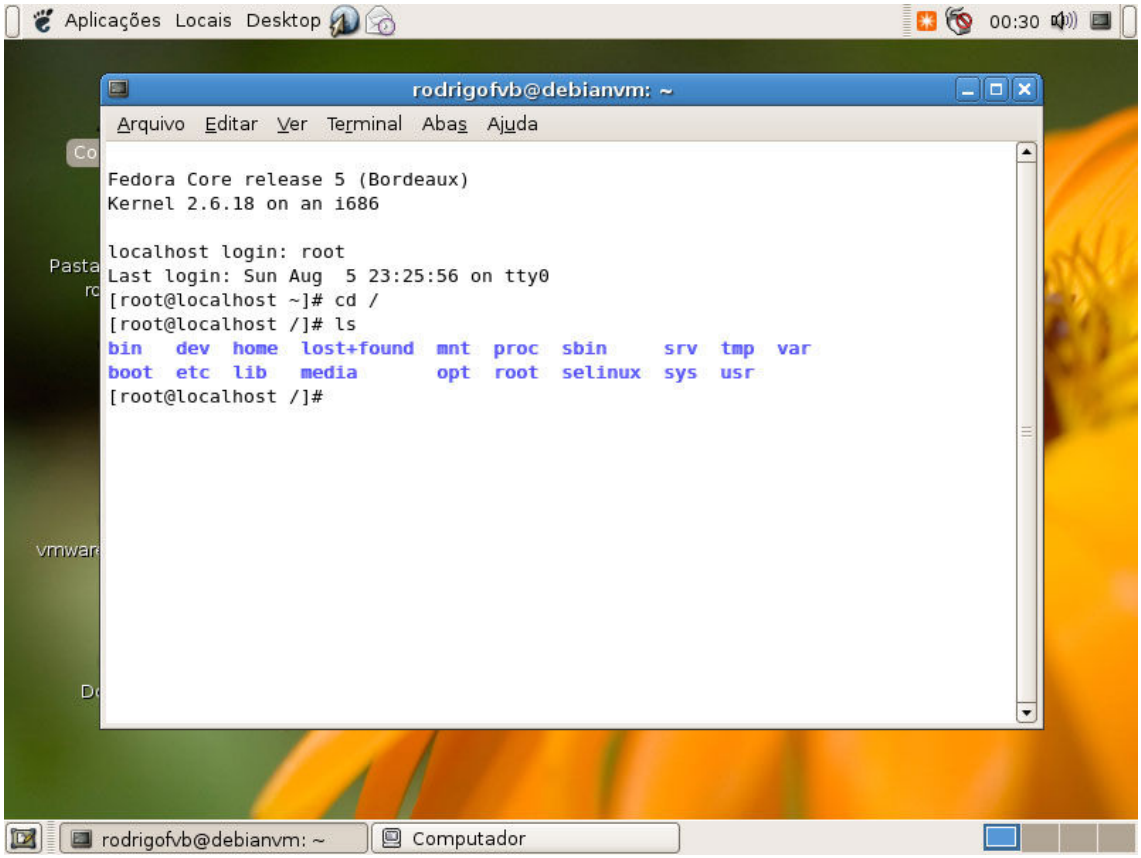


Figura 4.13 - Lista de pastas do Fedora em um ambiente completamente isolado do sistema anfitrião

CAPÍTULO V

VIRTUALIZAÇÃO NO NÍVEL DE SISTEMAS OPERACIONAIS

5.1 Introdução

Virtualização no nível de sistema operacional é uma tecnologia de virtualização que virtualiza servidores em uma subcamada do sistema operacional. Ele pode ser descrito como um particionamento de um servidor físico em múltiplas partições menores, e cada partição é isolada das demais e do sistema anfitrião.

A arquitetura da virtualização no nível de sistema operacional possui um baixo *overhead* que ajuda a maximizar (de forma eficiente) o uso de recursos do servidor. Como envolve apenas um *kernel*, este tipo de virtualização introduz um *overhead* quase insignificante e permite executar centenas de servidores virtuais isolados entre si, em um mesmo *hardware*, diferentemente do que acontece com outras técnicas (emulação, virtualização total ou paravirtualização) que não podem executar muitas máquinas virtuais por causa do alto *overhead* característico delas. Em con*trap*artida, a virtualização no nível de sistema operacional não permite executar diferentes sistemas operacionais (isto é, *kernels* diferentes), ainda que diferentes bibliotecas e distribuições sejam possíveis.

Comparando com soluções de virtualização por *hardware* e *software* como, por exemplo, o IBM's LPAR, a virtualização no nível de sistema operacional tem o benefício de executar em um *hardware* de baixo custo.

5.2 Solaris Containers (Zones)

Containers é a tecnologia de virtualização da Sun presente nativamente no Solaris 10 que é resultante da adição de inovações feitas no sistema operacional desde a versão 8. Solaris Containers compreende duas tecnologias:

- Solaris Zones, que provê um ambiente virtualizado que possui seu próprio nome de *host*, endereço de IP, usuários, sistema de arquivos, e isolamento de aplicações.
- Gerenciamento de recursos, que otimiza a carga de trabalho através de uma melhor distribuição dos recursos do sistema.

Zona (ou z*one*) é uma abstração do sistema operacional que provê um ambiente protegido para execução de aplicações. Cada zona é um ambiente virtualizado e tem sua própria identidade, que ficam separadas da subcamada do *hardware*. Como resultado, cada zona comporta-se como se estivesse executando seu próprio sistema operacional. Todas as zonas acessam o sistema concorrentemente, compartilhando os recursos do *hardware*. Em outras palavras, podemos dizer que, as zonas são máquinas virtuais que executam como instâncias do sistema operacional anfitrião (Solaris).

Devido ao fato do Solaris Containers ser independente da subcamada do *hardware*, os serviços podem ser recriados em diversos sistemas conforme a necessidade. Cada aplicação executa em seu próprio ambiente privado e muitas aplicações podem ser testadas e desenvolvidas em um simples servidor sem que isto afete outros sistemas. Caso o resultado seja indesejado, a zona pode ser facilmente destruída e recriada.

As zonas podem ser de dois tipos (figura 5.01):

- Zona global: é a instância principal do Solaris, todos os processos executam na zona global.
- Zonas não-globais: são ambientes virtuais opcionais que podem ser criados para hospedar aplicações (é a máquina virtual).

As principais características das zonas do Solaris são [VICTOR]:

- Múltiplas instâncias do sistema operacional.
- Alocação de recursos, incluindo CPU, memória física, banda de rede, e outros, baseado na carga de trabalho.

- Atribuição de ambientes virtuais às zonas, permitindo cada ambiente ter suas próprias configurações.
- Sistemas de arquivos privados ou compartilhados.
- Cada zona pode ter seu próprio endereço IP.
- As zonas podem ser criadas ou iniciadas manualmente, programaticamente ou automaticamente quando o sistema for iniciado.
- Possibilidade de desligar uma zona de dentro de outra ou da zona global.
- Possibilidade de usar pacotes de atualização para modificar a zona global, um subconjunto de zonas, ou todas as zonas.

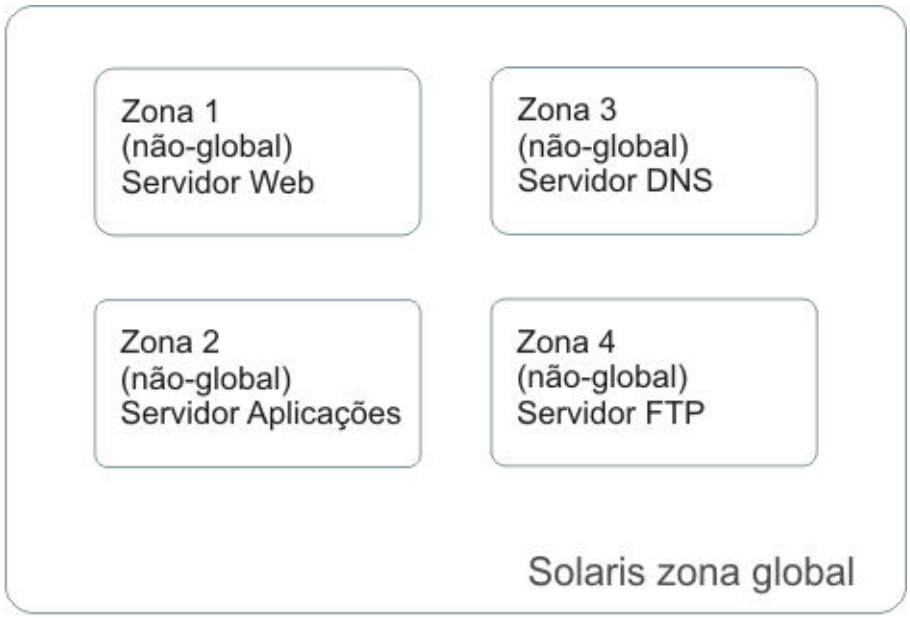


Figura 5.01 - Exemplo de aplicações isoladas nas zonas não-globais do Solaris

5.2.1 Gerenciamento de Memória

A memória é gerenciada pela zona global. Um benefício disto é que geralmente os executáveis usados e bibliotecas, podem ser carregados na memória uma vez e compartilhados entre as zonas, economizando consideravelmente os recursos de memória. Não está incluído no Solaris Containers a habilidade de limitar a memória disponível para zonas, entretanto um *daemon* do Solaris (rcapd) pode ser usado para limitar a quantidade de memória que cada zona pode usar. Quando um processo alcança seu limite de memória, o rcapd *daemon* força-o a enviar páginas para o dispositivo de *swap*.

5.2.2 Sistema de Arquivos

O Solaris Containers suporta o uso de múltiplos tipos de armazenamento como *Direct Attached Storage* (DAS[4]), *Network Attached Storage* (NAS[5]) e *Storage Area Network* (SAN[6]) e múltiplos tipos de sistema de arquivos.

É possível compartilhar sistema de arquivos para que os mesmos dados sejam utilizados por diferentes zonas.

Existem dois modelos de *layout* do arquivo do sistema operacional para uma zona não-global, que são:

- *Whole root*: uma zona *whole root* é uma cópia completa do sistema operacional no disco. Um processo executando em uma *whole root* pode "ver" só os arquivos nesta cópia, bem como os arquivos criados por usuários nesta zona. O superusuário (*root*) pode instalar novas versões do sistema e biblioteca de usuários. Este modelo oferece flexibilidade, todavia requer mais esforço de administração.
- *Sparse root:* uma zona *sparse root* não inclui sua cópia privada dos arquivos de sistema operacional e bibliotecas (não possui cópia dos diretórios /usr, /sbin, /platform e /lib). Os programas em zonas não-globais utilizam os arquivos de sistema da zona global. Este é o modelo padrão para zonas e economiza memória física e espaço em disco, além de simplificar a administração.

5.2.3 Configurações de Rede

As configurações de rede das zonas sempre são feitas a partir da zona global. Ao criar cada zona, deve-se definir as configurações de rede, porém estas configurações podem ser alteradas a qualquer momento pelo administrador, mesmo que as zonas estejam em execução.

[4] DAS é definido por dispositivo de armazenamento que está diretamente conectado ao computador

[5] NAS é caracterizado pela habilidade de prover acesso a dados através de uma rede.

[6] SAN são redes de dispositivos de armazenamento que provê acesso a dados aos servidores.

As comunicações entre zonas, por padrão, estão apenas disponíveis através das interfaces virtuais de rede. Entretanto, pode-se também configurar as zonas para se comunicarem entre si, mesmo que suas interfaces estejam em sub-redes diferentes.

5.2.4 Gerenciamento de Recursos

O gerenciamento de recursos é uma funcionalidade do ambiente do Solaris Containers que permite controlar como aplicações usam os recursos de sistema disponíveis.

Com o gerenciamento de recursos é possível:

- Alocar recursos computacionais, como por exemplo, tempo de processador.
- Monitorar as alocações que estão sendo utilizadas, e ajustá-las conforme necessário.
- Gerar relatórios de análises.

Existem no Solaris Containers três tipos de mecanismos de controle para gerenciamento de recursos, são elas: *Constraint,* que permite ao administrador ou desenvolvedor de aplicações definir os limites de consumo de recursos específicos para uma carga de trabalho. Com limites conhecidos, a modelagem de cenários de consumo de recursos torna-se um processo bastante simples. Os limites também podem ser usados para controlar aplicações com mau funcionamento que poderiam comprometer o desempenho do sistema. *Scheduling*, o qual permite que se faça uma seqüência de alocações em intervalos específicos. *Partitioning*, usado para vincular uma carga de trabalho a um subconjunto de recursos disponíveis no sistema. Este vínculo garante que uma fatia dos recursos esteja sempre disponível para determinada carga de trabalho [SUN 2004].

5.2.5 Teste do Solaris Zones

Os passos a seguir descrevem uma implementação de zona não-global numa instalação do Solaris 10 que fiz em meus testes:

1. Na zona global iniciei um console e nela defini uma zona não-global chamada ‘znrodrigo’.
2. Nesta zona atribuí o endereço IP ‘192.168.0.11’ ao adaptador de rede (que no caso era ‘pcn0’).
3. Instalei o sistema Solaris e inicializei-o, conforme Figura 5.02.

Os comandos para estes procedimentos foram:

```
# zonecfg –z znrodrigo
zonecfg:znrodrigo> create
zonecfg:znrodrigo> set zonepath=/zones/zone_roots/znrodrigo
zonecfg:znrodrigo> add net
zonecfg:znrodrigo:net> set address=192.168.0.11
zonecfg:znrodrigo:net> set physical=pcn0
zonecfg:znrodrigo> end
zonecfg:znrodrigo> commit
zonecfg:znrodrigo> exit
# zoneadm –z znrodrigo install
```

Após a nova zona ser inicializada, efetuei o *login* nessa nova zona através de um console com o comando:

```
# zlogin –C znrodrigo
```

Na figura 5.03 pode-se ver o sistema de arquivos da zona, obtido com o comando:

```
# ls -l
```

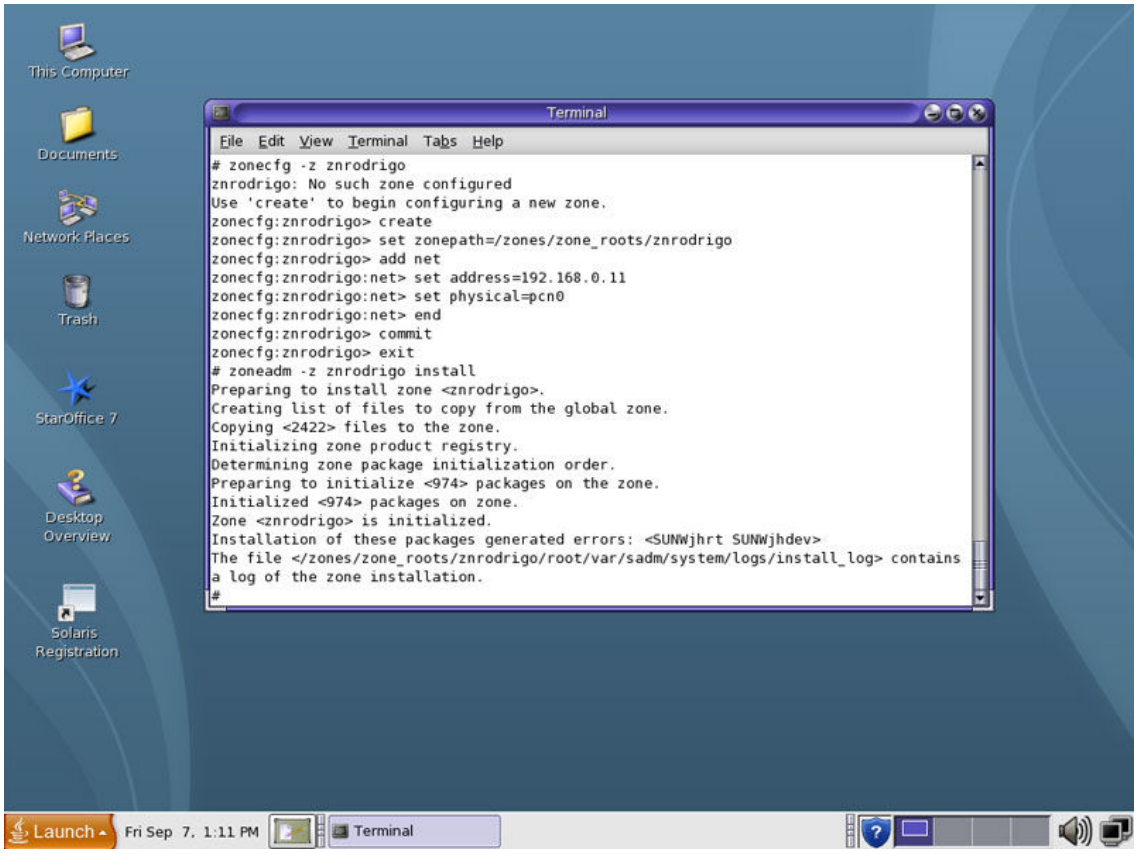


Figura 5.02 - Criação de uma zona chamada ‘znrodrigo’ no Solaris

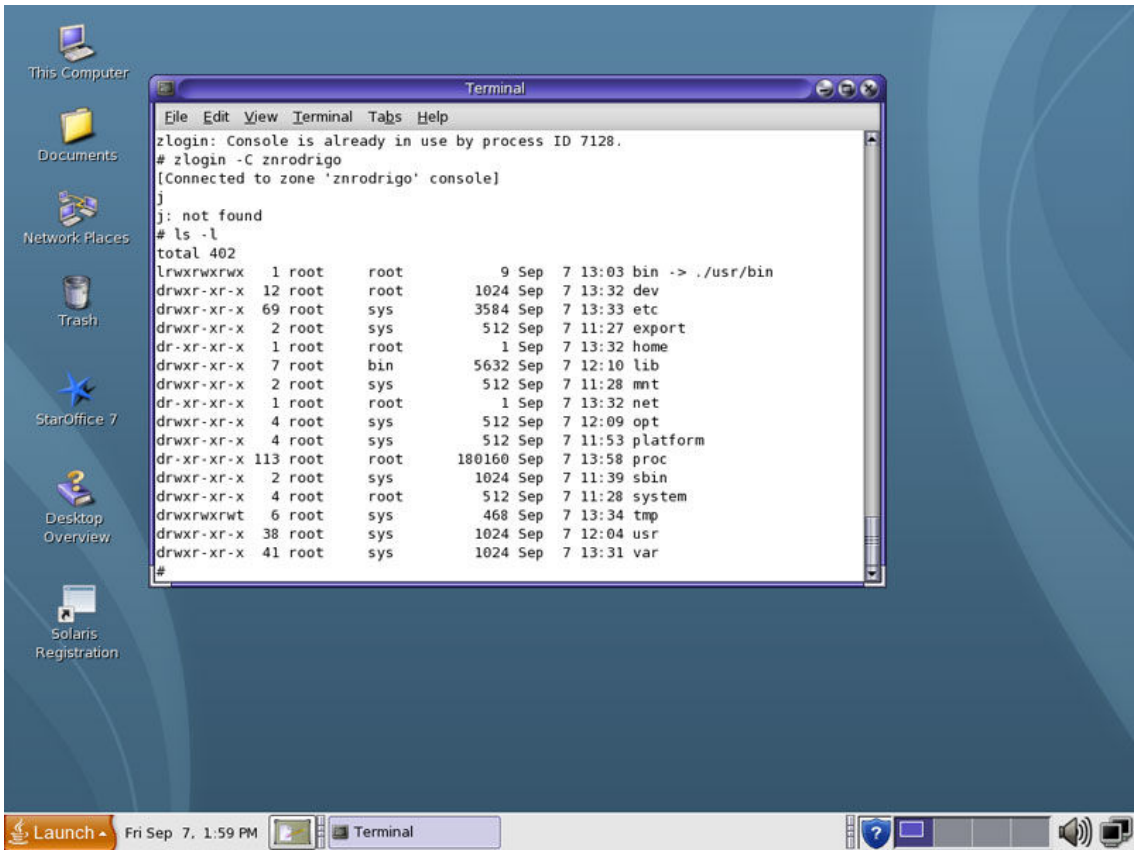


Figura 5.03 - Sistema de arquivos da zona não-global

Por fim, na figura 5.04 podemos observar um console da zona não–global “znrodrigo” e o console da zona global chamada “vmsolaris1”.

Com isto, pude concluir que criar e administrar zonas no Solaris é uma tarefa fácil, e cada zona criada, pode possuir características como um sistema de arquivos e um endereço IP próprio, além de outros elementos.

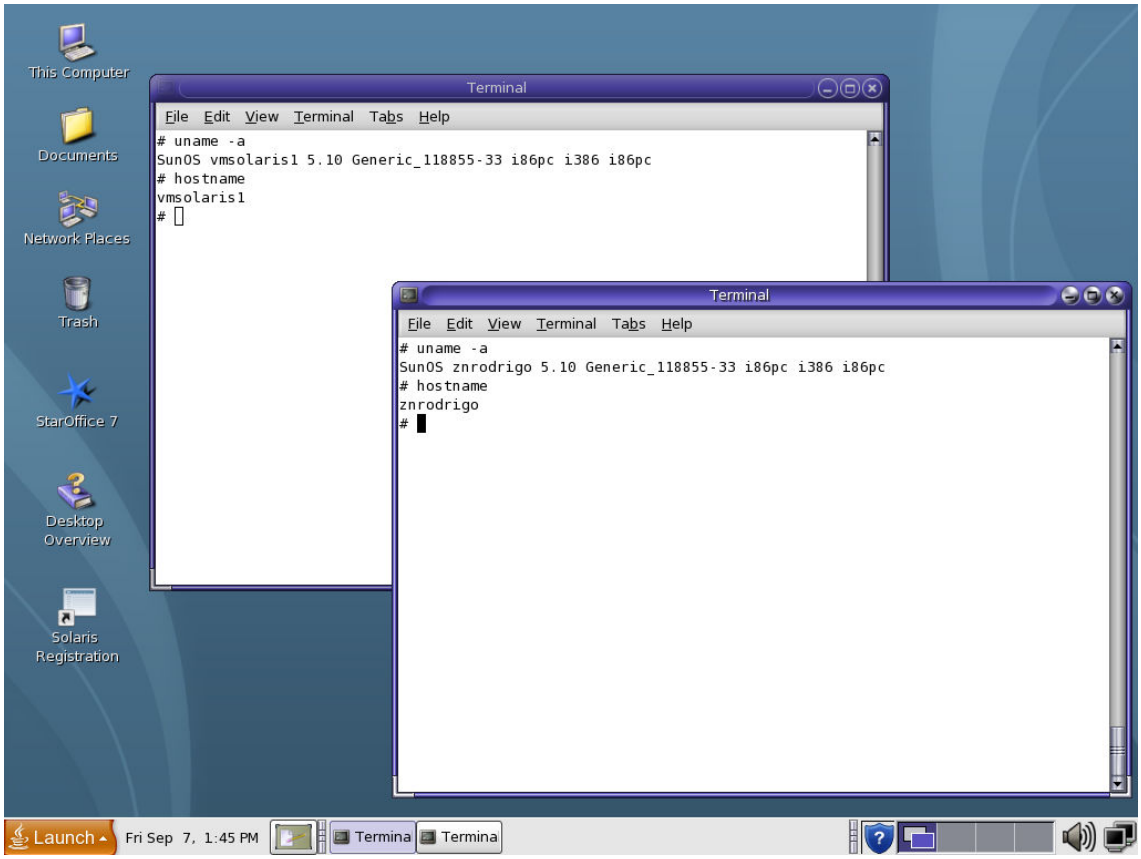


Figura 5.04 - Zona global ‘vmsolaris1’ e zona não-global ‘znrodrigo’ executando simultaneamente

5.3 FreeBSD Jails

Jails (prisões) são partições nativas do ambiente do sistema operacional FreeBSD, o qual podem abranger processos, sistema de arquivos e recursos de rede. Esta funcionalidade é proveniente do modelo de segurança do sistema Unix, permitindo múltiplos usuários e um superusuário (*root*) privilegiado em cada partição, enquanto limita o escopo das atividades do superusuário na mesma. O administrador da máquina

FreeBSD pode particionar a máquina em prisões separadas, e prover acesso de superusuário (*root*) em cada uma delas, sem perder o controle sobre todo o ambiente.

Quando um processo está aprisionado (dentro de uma jail), os seus processos filhos[7] também são aprisionados. Um processo pode executar em apenas uma prisão, e após a criação do processo, ele não pode mais deixar esta prisão.

A chamada de sistema `jail` cria prisões, e pode ser implementada de diversas maneiras, porém de uma forma usual, a configuração cria uma completa instalação do FreeBSD para cada prisão, o que inclui cópias de todos os binários relevantes do sistema bem como arquivos de dados e seu próprio diretório /etc (figura 5.05). Esta configuração maximiza a independência de várias prisões e reduz as chances de interferência entre prisões quando possível, especialmente quando é desejável prover acesso de *root* (de dentro de uma prisão) para um usuário menos confiável, por exemplo.

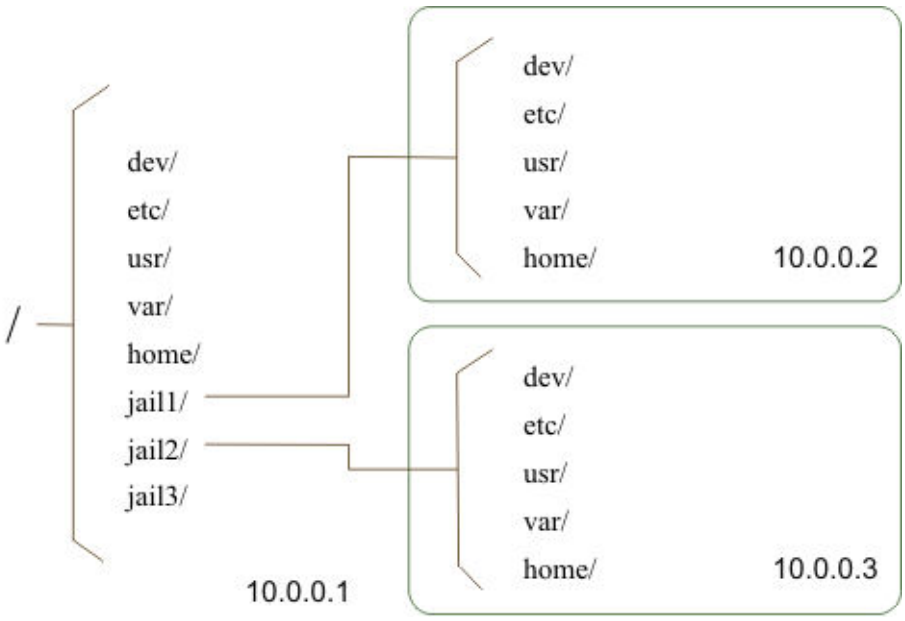


Figura 5.05 - Diagrama do FreeBSD com duas "prisões" configuradas da forma usual

Uma vez que uma prisão é criada, ela somente endereça um sistema de arquivos exclusivo a ela, e como os processos não podem manipular arquivos que não endereçam, a integridade e confidencialidade de arquivos que estão fora da prisão estão garantidas.

[7] Processos criados a partir de um processo pai.

Atualmente somente um simples endereço IP pode ser alocado para cada prisão e toda comunicação da prisão está limitada ao endereçamento IP. Os processos aprisionados não podem utilizar outros endereços IP para conexões de entrada e saída.

Um processo executando com privilégios de *root* e confinado em uma prisão não pode:

- Modificar o *kernel* atual.
- Modificar configurações de rede.
- Montar e desmontar sistemas de arquivos.
- Criar nós de dispositivos.
- Alterar níveis de segurança de arquivos.
- Acessar recursos de rede não associados com a prisão.

Como visto, a funcionalidade FreeBSD jails provê um mecanismo simples de particionamento, permitindo a delegação de permissões administrativas dentro das máquinas virtuais e também limita a interação entre processos e arquivos, recursos de rede, e operações de privilégios [KAMP e WATSON].

5.3.1 Teste do FreeBSD Jails

Para implementação da prisão, utilizei o FreeBSD versão 6.2, o qual instalei a interface gráfica KDE. O *hardware* constituiu-se de um PC de arquitetura simples. É importante salientar que dentre os sistemas virtuais testados por mim, o FreeBSD Jails foi o que apresentou maior grau de dificuldade na criação de máquinas virtuais, necessitando editar vários arquivos do sistema operacional FreeBSD.

Nessa instalação foram definidos os seguintes dados para o sistema anfitrião:

- Hostname: vmfreebsd
- IP: 192.168.188.129
- Interface de rede: lnc0

A seguir, estipulei os seguintes dados para o novo sistema virtual:

- Hostname: vm1

- IP: 192.168.188.130
- Caminho para os arquivos do sistema virtual: /usr/jails/vm1

Esses foram os passos utilizados para configurar a prisão:

1. Criação da pasta e geração dos arquivos para o sistema virtual:

```
# mkdir /usr/jails/vm1
# cd /usr/src
# make world DESTDIR=/usr/jails/vm1
# cd etc
# make distribution DESTDIR=/usr/jails/vm1
```

2. Edição do arquivo /etc/rc.conf, acrescentando os dados da nova prisão:

```
inetd_enable=”YES”
inetd_flags="-wW -a 192.168.0.129"

jail_enable="yes"
jail_list="vm1"        # apenas uma prisão definida (vm1)

jail_vm1_rootdir="/usr/jails/vm1/"
jail_vm1_hostname="vm1"
jail_vm1_ip="192.168.188.130"
jail_vm1_exec="/bin/sh /etc/rc"
jail_vm1_devfs_enable="YES"
jail_vm1_fdescfs_enable="YES"
jail_vm1_procfs_enable="YES”
jail_vm1_fstab=""
jail_vm1_flags="-l -U root"
```

3. Criação do arquivo fstab vazio e cópia do arquivo resolv.conf do sistema anfitrião para a pasta /etc do sistema virtual:

```
# touch /usr/jails/vm1/fstab
# cp /etc/resolv.conf /usr/jails/vm1/etc
```

4. Inicialização da nova prisão, conforme visto na figura 5.06:

```
# ifconfig lnc0 inet alias 192.168.188.130/24
# mount_devfs devfs /usr/jails/vm1/dev
# mount_procfs procfs /usr/jails/vm1/proc
# jail /usr/jails/vm1 vm1 192.168.188.130 /bin/sh
```

Na figura 5.07, vemos um teste de ping feito com sucesso entre o sistema anfitrião e a nova prisão, o comando utilizado foi:

```
# ping 192.168.188.130
```

Com todos esses testes pude concluir que o FreeBSD jails, assim como o Solaris Zones, também é uma grande ferramenta para isolamento de processos, e com a vantagem de já estar disponível no kernel do FreeBSD. Ele também disponibiliza um novo sistema de arquivos (uma cópia do sistema anfitrião) bem como um novo endereço IP para cada prisão.

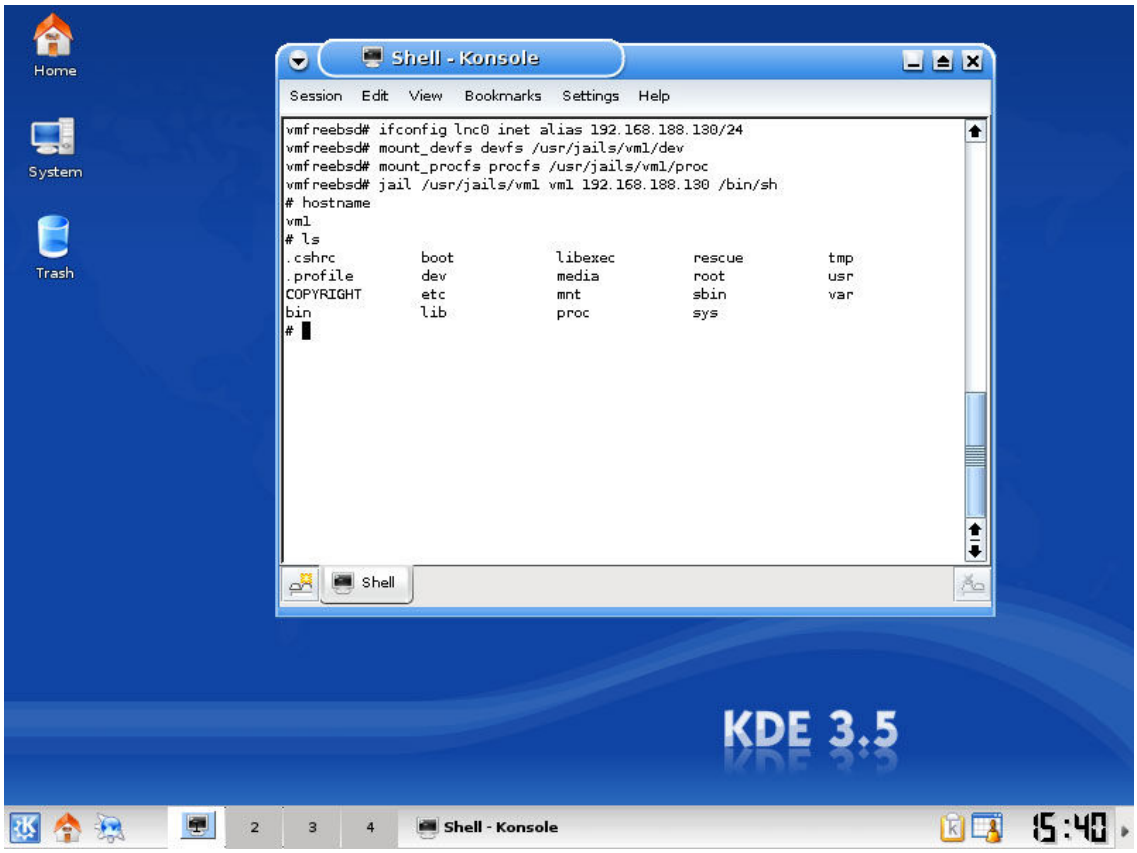


Figura 5.06 - Inicialização da nova prisão em um console e listagem do novo sistema de arquivos

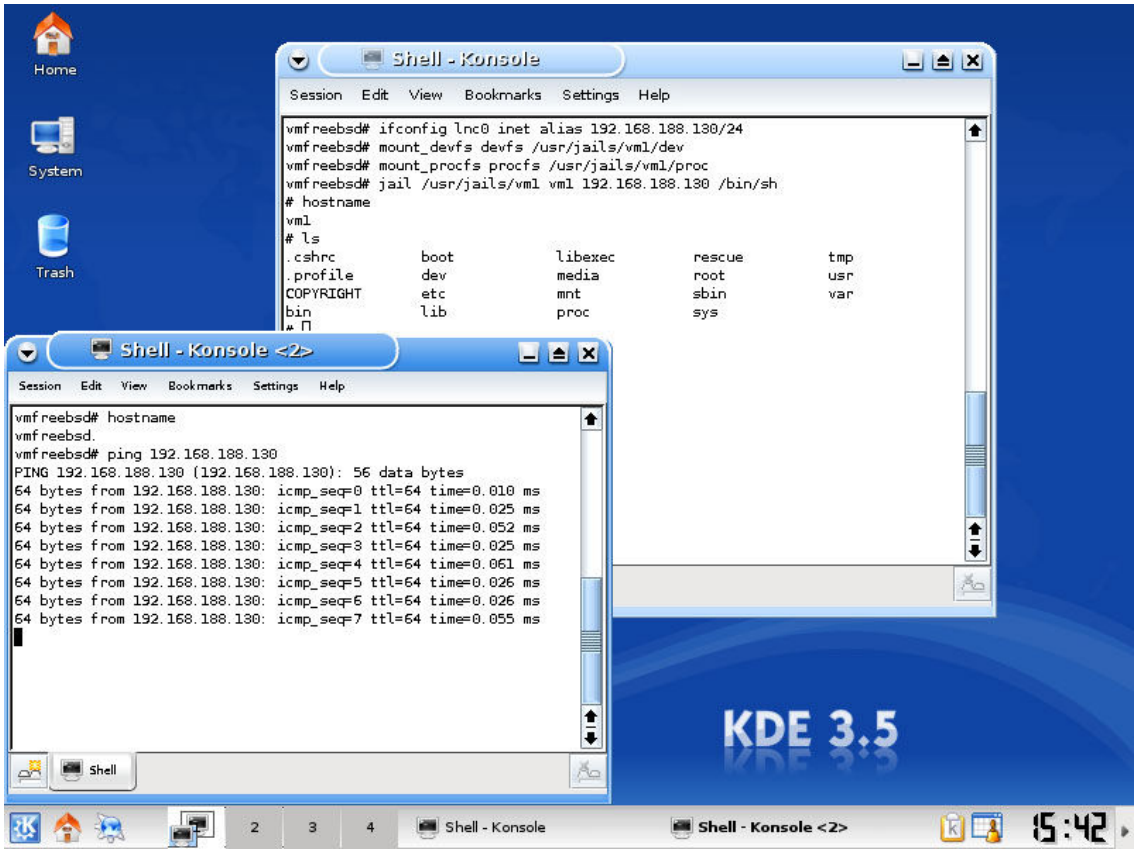


Figura 5.07 - Teste de ping partindo do sistema anfitrião com destino ao IP da nova prisão

CAPÍTULO VI

VIRTUALIZAÇÃO POR EMULAÇÃO

6.1 Introdução

Esta técnica consiste em fazer uso de emuladores para prover a utilização de sistemas operacionais em máquinas virtuais.

Esta técnica é especialmente útil em casos onde é preciso utilizar sistemas operacionais que não foram escritos para a arquitetura do *hardware* da máquina anfitriã, ou executar um simples sistema operacional sobre outro diferente. A desvantagem desta técnica consiste na necessidade de executar uma nova instância do emulador para cada máquina virtual criada. Com isso, para aplicações que requeiram múltiplos sistemas operacionais diferentes, os sistemas de virtualização que utilizam camada de abstração do *hardware* são os mais indicados.

6.2 QEMU

O QEMU é um *software* emulador de processador que utiliza a técnica de tradução dinâmica. Como já foi visto, esta técnica traduz partes do código para que o processador execute as instruções. O QEMU foi escrito por Fabrice Bellard e é um *software* livre (cuja maior parte está sob licença GNU LGPL[8]).

O QEMU tem dois modos de operação:

- Emulação total do sistema: neste modo o QEMU emula um sistema completo (x86, por exemplo) que pode ser com um ou mais processadores e vários

[8]GNU Lesser General Public License, é uma licença de software escrita com o intuito de ser um meio-termo entre a GPL e licenças mais permissivas como a licença BSD.

periféricos. Assim ele pode ser usado para executar diferentes sistemas operacionais.

- Emulação de modo usuário (somente Linux): Neste modo, ele pode executar processos Linux compilados de uma CPU em outra CPU. Com isso, um programa compilado para um processador PowerPC pode ser executado em x86 e vice-versa.. É principalmente usado para testar o resultado de compiladores, ou para testar o emulador de CPU sem que seja necessário iniciar uma máquina virtual completa.

Principais características:

- Emula arquiteturas como x86 32 e 64, PowerPC, MIPS, ARM e Sparc 32 e 64 (todavia as arquiteturas x86 e PowerPC são melhor suportadas).
- Emula dispositivos como vídeo VGA, porta serial, mouse e teclado PS/2, disco rígido IDE, adaptador de rede NE2000.
- Suporta auto-emulação, isto é, é possível executar o QEMU de dentro do QEMU (embora isto não seja indicado devido a uma perda acentuada de desempenho).
- Não requer alterações no sistema anfitrião e convidado.
- Controle remoto da máquina emulada por meio de um servidor VNC.
- Pode salvar e recuperar o estado de uma máquina, bem como de seus programas em execução.
- A versão para sistemas anfitrião Windows é experimental (embora o QEMU suporte bem o Windows dentro de uma máquina virtual).

No gerenciamento de memória, o QEMU usa uma chamada de sistema (*mmap)* do sistema anfitrião para simular uma *Memory Management Unit* (MMU) alvo. O QEMU também suporta um *software* MMU. Neste modo, a tradução do endereçamento físico-virtual é feita a cada acesso à memória. Na emulação de processos do usuário, nenhuma simulação MMU é feita porque o QEMU pressupõe que o mapeamento da memória do usuário é feito pelo sistema anfitrião [BELLARD 2005].

Sendo o QEMU um emulador, e necessário fornecer à máquina virtual todas as instruções que ela utiliza, o que afeta diretamente seu desempenho. O QEMU é quatro vezes mais lento do que sistemas nativos em operações de inteiros, e dez vezes mais

lento em operações de ponto flutuante. Contudo ele é aproximadamente trinta vezes mais rápido que o Bochs, que será visto mais adiante [BELLARD 2007].

6.2.1 Teste do QEMU

Em meu experimento, fiz uma instalação QEMU baixando os pacotes através da Internet em um sistema Linux Debian 4.0 e instalei o sistema operacional Windows XP como sistema convidado. Esta instalação foi feita através de um CD de instalação do Windows XP em um sistema de arquivos virtual que chamei de 'hdv1' definido na propriedade da criação do processo QEMU. O *hardware* utilizado foi um PC de configuração simples.

Os comandos utilizados foram os a seguir:

```
# apt-get install qemu
# dd if=/dev/zero of=hdv1 bs=1024 count=1400000
# qemu -cdrom /dev/cdrom -boot d hdv1
# qemu -boot c hdv1
```

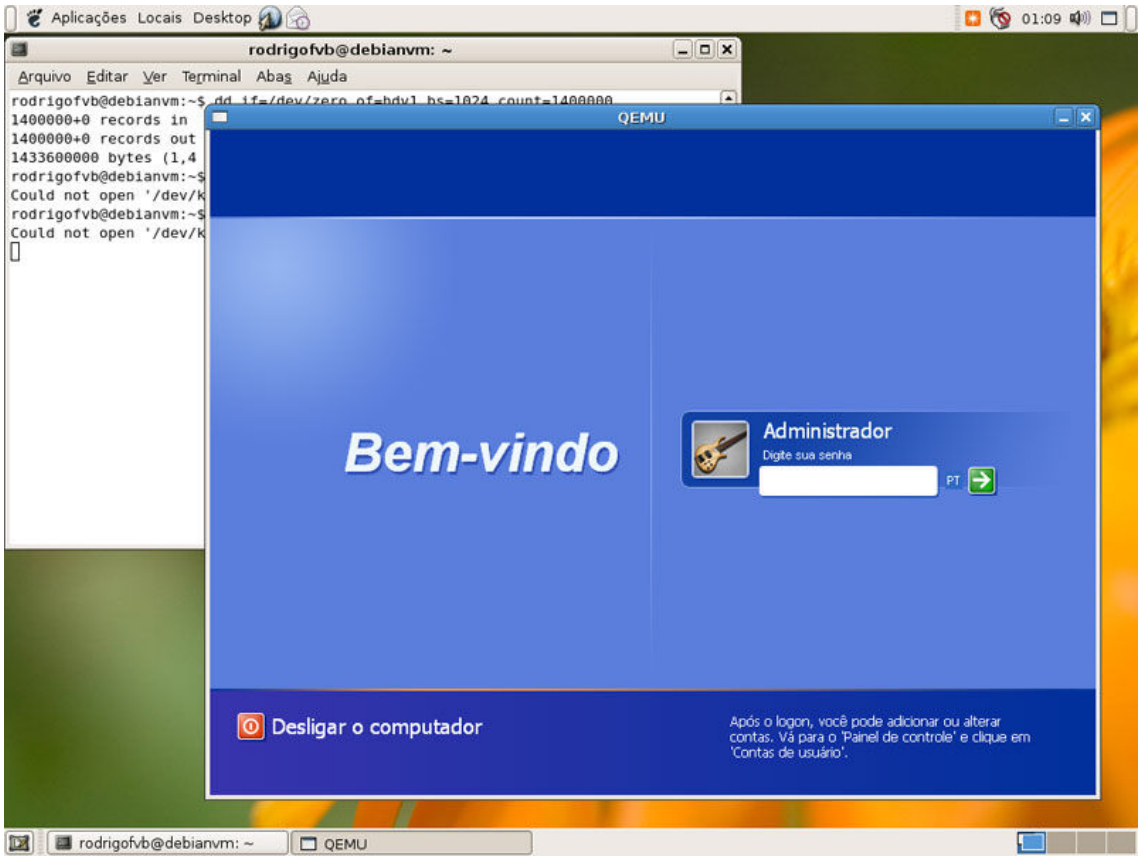


Figura 6.01 - Sistema Linux Debian com QEMU executando Windows

Nesse teste, o sistema Windows XP funcionou normalmente, porém o desempenho do sistema mostrou-se bastante reduzido mesmo com apenas um processo QEMU em execução, diferentemente do que já havia sido experimentado em outros métodos de virtualização testados anteriormente.

6.3 Bochs

O Bochs é um programa que simula totalmente um computador de arquitetura x86. Ele foi desenvolvido Kevin Lawton em 1994 para ser um produto comercial, todavia, foi adquirido em março de 2000 pela empresa MandrakeSoft (atualmente Mandriva) que o tornou *open source*, sob a licença GNU LGPL.

O Bochs suporta emulação de CPU (podendo funcionar como uma CPU 386, 486, Pentium e AMD64, incluindo as instruções MMX, SSE/SSE2/SSE3 e 3DNow!), memória, disco, vídeo, rede, BIOS e os periféricos mais comuns.

Foi escrito na linguagem C++ e desenvolvido para executar em diferentes plataformas anfitriãs, como x86, PPC, Alpha, Sun e MIPS. Difere de outros emuladores pelo fato de não depender totalmente de instruções nativas da máquina anfitriã, simulando por meio do *software* cada instrução x86. Por este motivo, ele pode, por exemplo, executar um sistema operacional Windows em um *hardware* Alpha ou Sparc [LAWTON et al].

6.3.1 Teste do Bochs

Realizei uma instalação da versão do Bochs para Windows em um sistema operacional Windows XP e em seguida, instalei o sistema MS-DOS como convidado.

No teste os comandos do MS-DOS funcionaram sem problemas, todavia o sistema apresentou um desempenho muito reduzido, o que me fez concluir que o Bochs não é a melhor opção para emular sistemas operacionais escritos para uma mesma plataforma, todavia, pode ser uma opção interessante para emular sistemas operacionais em

arquiteturas de *hardware* para os quais eles não foram escritos originalmente (por exemplo: Windows em plataforma Sparc). A figura 6.02 ilustra o teste supracitado.

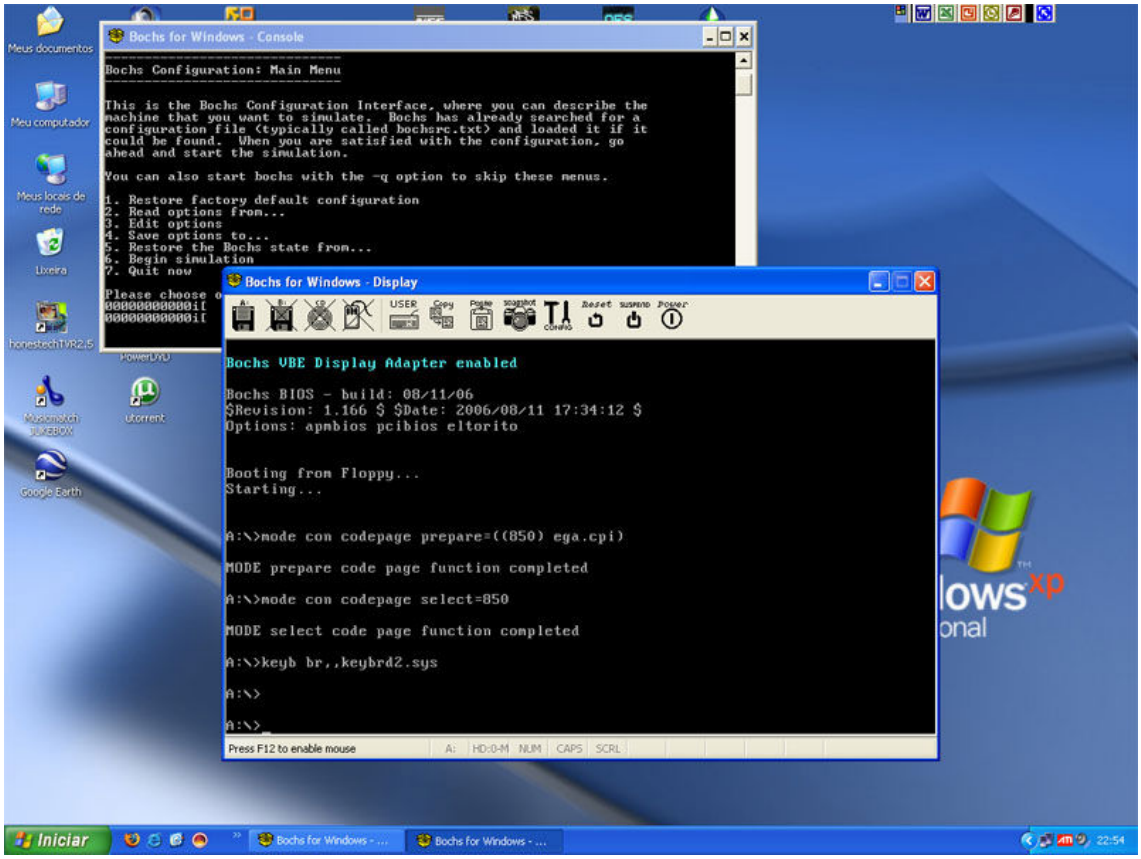


Figura 6.02 - Bochs para Windows executando MS-DOS

CAPÍTULO VII

VIRTUALIZAÇÃO NO NÍVEL DE APLICAÇÃO

7.1 Introdução

Na virtualização no nível de aplicação, a máquina virtual não provê um ambiente completo para execução de sistemas operacionais. Na verdade a máquina virtual é um componente chave para execução de determinadas aplicações, com a função de receber um programa em uma linguagem e fazer uma compilação do programa em tempo de execução. Essa técnica é conhecida como compilação *just-in-time* ou JIT. Desse modo, não há necessidade do programa ser compilado previamente para execução em plataformas específicas (como nas aplicações tradicionais).

O principal objetivo deste modelo é de garantir uma maior portabilidade para os programas, uma vez que um mesmo programa pode ser executado em qualquer plataforma que exista uma máquina virtual escrita para ela.

7.2 Java Virtual Machine

O Java Virtual Machine (JVM ou máquina virtual Java) é a pedra fundamental da linguagem Java da Sun Microsystems. Ele é o componente da tecnologia responsável pela independência entre o *hardware* e o sistema operacional, o pequeno tamanho do código compilado, e a habilidade de proteger usuários contra programas maliciosos.

Ele é uma abstração da máquina, porém sua implementação não assume qualquer tecnologia em particular (apesar de padronizada pela Sun), *hardware* anfitrião, ou sistema operacional anfitrião.

Uma curiosidade é que o primeiro protótipo de implementação do JVM, feito pela Sun, emulava um conjunto de instruções da máquina virtual Java em um dispositivo *handheld*.

O JVM não "conhece" a linguagem de programação Java, ele só executa um formato de arquivo binário, o formato de arquivo "*.class*" que contém instruções para a máquina virtual Java, chamado de *bytecodes* [LINDHOLM e YELLIN].

O JVM interpreta o arquivo de *bytecode* e se encarrega de executar os comandos no sistema operacional onde o programa está executando. O *bytecode* pode ser interpretado por qualquer máquina virtual Java, executando em diversos sistemas operacionais como Linux, Windows e Solaris, ou qualquer outro sistema operacional que possua uma implementação de JVM.

Ele também não permite que um programa Java acesse diretamente os recursos de *hardware*, protegendo o sistema de operações perigosas, como acesso a regiões protegidas da memória ou áreas do disco rígido.

7.3 Microsoft .Net CLR

O CLR (*Common Language Runtime*) é a máquina virtual do Microsoft .Net, cujo objetivo é prover um ambiente abstrato para execução de aplicações de forma independente da plataforma de uso.

As implementações da Microsoft para o CLR seguiram o padrão de arquitetura ECMA-335 CLI (*Common Language Interface*), que é um conjunto de especificações para execução de aplicações escritas em diversas linguagens de programação diferentes.

O Microsoft .Net compila todo código escrito nas linguagens do ambiente .Net (por exemplo: VB.Net, C#.Net, Asp.Net) em uma linguagem intermediária que é comum à todas as linguagens. Esta linguagem intermediária chama-se *Microsoft Intermediate Language* (MSIL), e é o código que é interpretado pela máquina virtual CLR. Em tempo

de execução, o compilador *just-in-time* converte o código MSIL em código nativo para o sistema operacional [BOCCATO et al].

O CLR além de prover um nível abstração que permite que os desenvolvedores de *software* ignorem detalhes de sistemas operacionais e *hardware*, ele também provê outras características importantes como:

- É orientado a objetos e suporta herança simples e polimorfismo.
- Possui um conjunto de classes que encapsulam a maioria da funcionalidade da API do Windows e outras tecnologias, como o XML.
- Proporciona execução virtual de código e tratamento de memória automática.
- Possui coletor de lixo (*garbage collector*).
- Uma vez escrito e compilado para MSIL, uma aplicação .NET administrada pode ser executada em qualquer plataforma que suporte CLR.
- O CLR pode verificar a segurança de tipo de todos os seus códigos, eliminando muitos erros comuns de programação.
- Tratamento de exceções e opções avançadas de depuração de aplicações.

CAPÍTULO VIII

BIBLIOTECAS DE INTERFACE DO USUÁRIO

8.1 Introdução

É uma prática comum dos desenvolvedores de *software* escrever sistemas e aplicações utilizando-se de um conjunto de API (*Application Programming Interface*) fornecida pelo sistema operacional e acessadas pelo usuário por meio de bibliotecas. A virtualização nesta forma é conseguida inserindo uma camada que captura as chamadas as API pela aplicação e emulando-as, com o intuito de que a aplicação execute em um sistema operacional diferente do qual foi originalmente projetada. Um bom exemplo de *software* que faz uma implementação deste tipo é o Wine.

8.2 Wine

O Wine é uma implementação da API do Windows sobre a plataforma Unix. É uma camada de tradução entre o Unix e um aplicativo para Windows, que intercepta as chamadas a API do Windows feita pelos aplicativos e converte-as em chamadas equivalentes próprias dos sistemas Unix. Segundo a Sun [SUN 2003], "o Wine é uma implementação da API do Windows no topo do *software* Unix e o sistema X Window".

Atualmente o Wine é um projeto *open source* sob a licença GNU LGPL e suporta aplicações escritas para Windows sobre as plataformas Linux, Solaris, Mac OS/X e FreeBSD. O projeto foi iniciado em 1993 por Bob Amstadt, como uma forma de suportar aplicativos do Windows 3.1 em Linux. Posteriormente o projeto foi entregue a Alexandre Julliard que o coordena até os dias atuais.

Podemos dizer que o Wine é um "carregador", o qual carrega e executa programas Windows e uma série de bibliotecas que emulam chamadas às funções da API do Windows.

Porque o Wine não provê um ambiente completo para execução de aplicativos (como um sistema operacional ou um *hardware* virtual) ele não considerado um emulador total. O próprio nome Wine reafirma isso, pois é um acrônimo recursivo de *Wine is not Emulator* (o Wine não é um emulador). Desse modo, ele permite que as aplicações convidadas executem com o máximo de desempenho, obtendo uma vantagem sobre a maioria das soluções de virtualização.

Contudo, o Wine ainda é um projeto em desenvolvimento e muitas aplicações ainda não funcionam por problemas de compatibilidade, principalmente aplicações mais recentes. Isso se deve principalmente ao fato de que algumas companhias de *software* não revelarem maiores detalhes das operações internas de seus sistemas operacionais e aplicações.

Para conhecimento, o Cygwin é um aplicativo similiar ao Wine, porém funciona de forma inversa, ele é uma implementação da API do Linux sobre a plataforma Windows.

8.2.1 Arquitetura do Wine

O funcionamento do Wine está fundamentalmente ligado à execução de um processo chamado *wineserver*. É o *wineserver* que gerencia a execução todos os processos `wine` e suas *threads*. Todos os objetos Win32[9] do cliente `wine` também são gerenciados pelo *wineserver*, e todos os clientes devem enviar requisições ao *wineserver* sempre que eles precisarem saber sobre qualquer descritor de arquivo Unix associado a um objeto Win32.

Os aplicativos Windows fazem freqüentemente chamadas as *Dynamic Link Library* (DLL[10]). Para isto o Wine contém um conjunto de implementações de DLL Quando uma aplicação precisa importar uma DLL, o Wine faz um busca em sua lista de DLL registradas.

[9] Win32: API das versões 32 bits do sistema operacional Windows.

[10] DLL: Bibliotecas compartilhadas de funções do Windows.

O Wine substitui completamente três das principais DLL do Windows (Kernel/Kernel32, GDI/GDI32, User/User32), cujas quais todas as outras DLL estão submetidas de alguma forma. O NTDLL é outra importante DLL de núcleo do Windows implementada no Wine [WINEHQ]. A figura 8.01 ilustra a arquitetura do Wine.

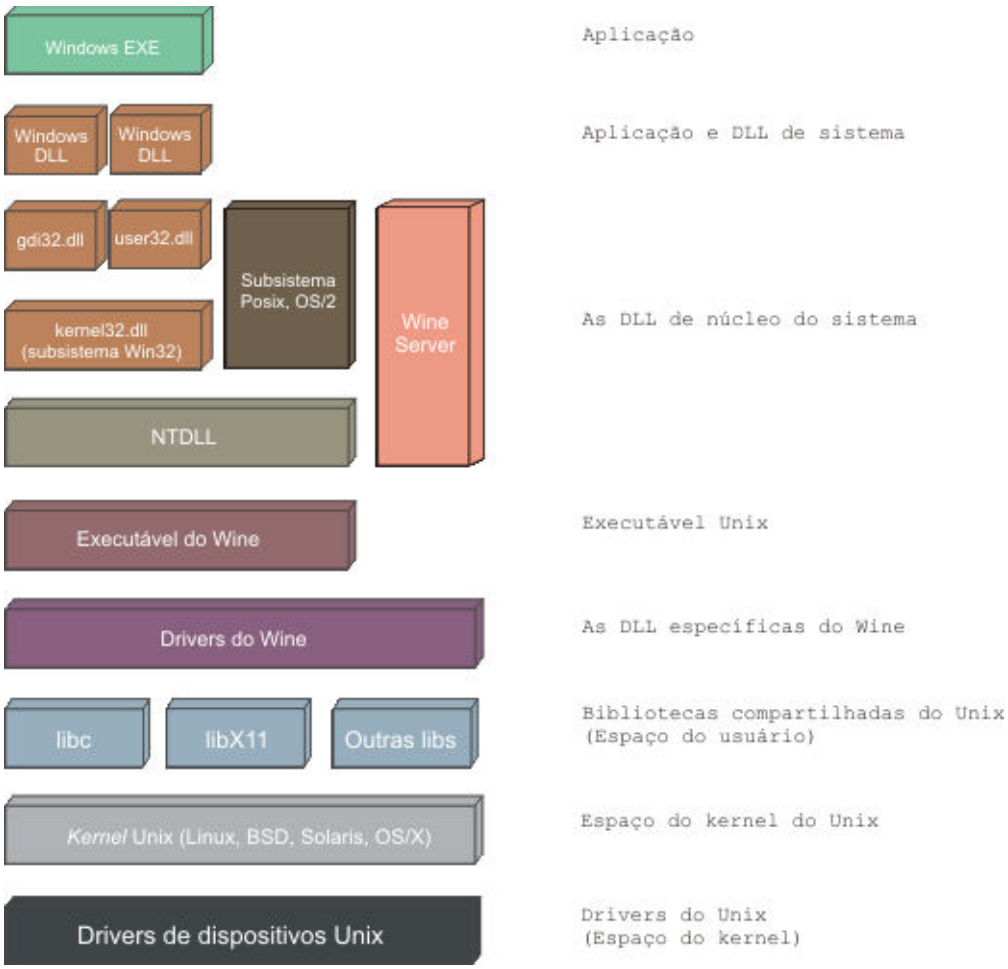


Figura 8.01 - Representação da arquitetura do Wine em camadas

8.2.2 Gerenciamento de memória

Cada processo no Wine tem seu próprio processo no sistema anfitrião e com isso seu próprio espaço de endereçamento.

8.2.3 *Drivers* Wine

O Wine não permite utilizar diretamente *drivers* Windows "debaixo" do Unix. Ele somente é capaz de prover acesso a um dispositivo específico se o mesmo for suportado pelo Unix, ou seja, no caso de existir um *driver* Unix para ele. O Wine também provê o acesso nos casos em que ele possui uma implementação que faça uma "ponte" entre a API do *driver* Windows e o *driver* Unix [WINEHQ].

8.2.4 Teste do Wine

Primeiramente baixei e instalei os pacotes do Wine com o comando:

```
# apt-get install wine
```

Logo após modifiquei as configurações padrões do Wine para emular o Windows XP com o comando:

```
# winecfg
```

A seguir instalei o *software* Adobe Reader 6.0 para Windows com o comando:

```
# wine "AdbeRdr60_ptb_full.exe"
```

A figura 8.02 mostra o *software* Adobe Reader 6.0 executando normalmente sobre o Linux com o Wine.

Fiz também uma instalação de um *software* conversor de vídeo *freeware*, conforme figura 8.03.

Com esses testes pude concluir que o Wine é uma excelente solução para executar aplicações Windows sobre o ambiente Linux sem que seja necessário virtualizar um sistema operacional completo. Todavia, o Wine ainda possui muitos problemas de compatibilidade, os quais pude constatar quando fiz tentativas de instalação de aplicativos mais complexos.

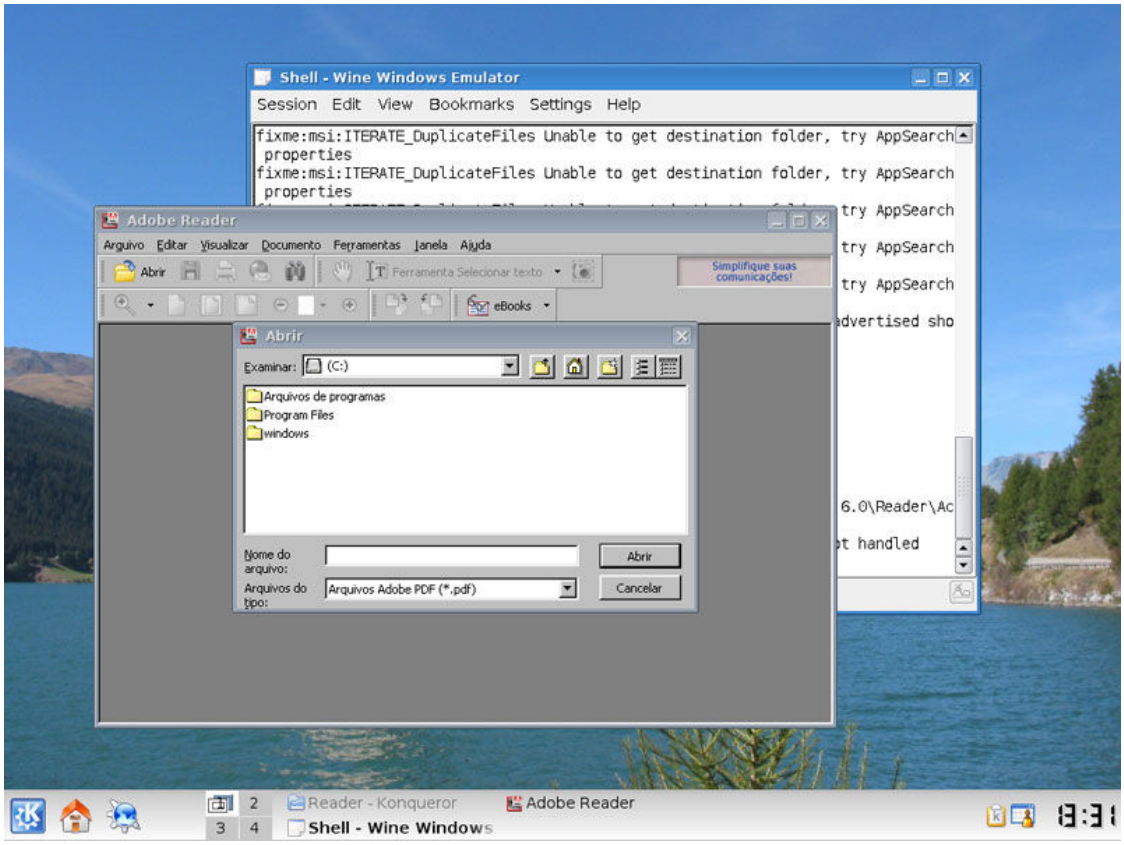


Figura 8.02 - Adobe Reader para Windows executando no Linux com o Wine

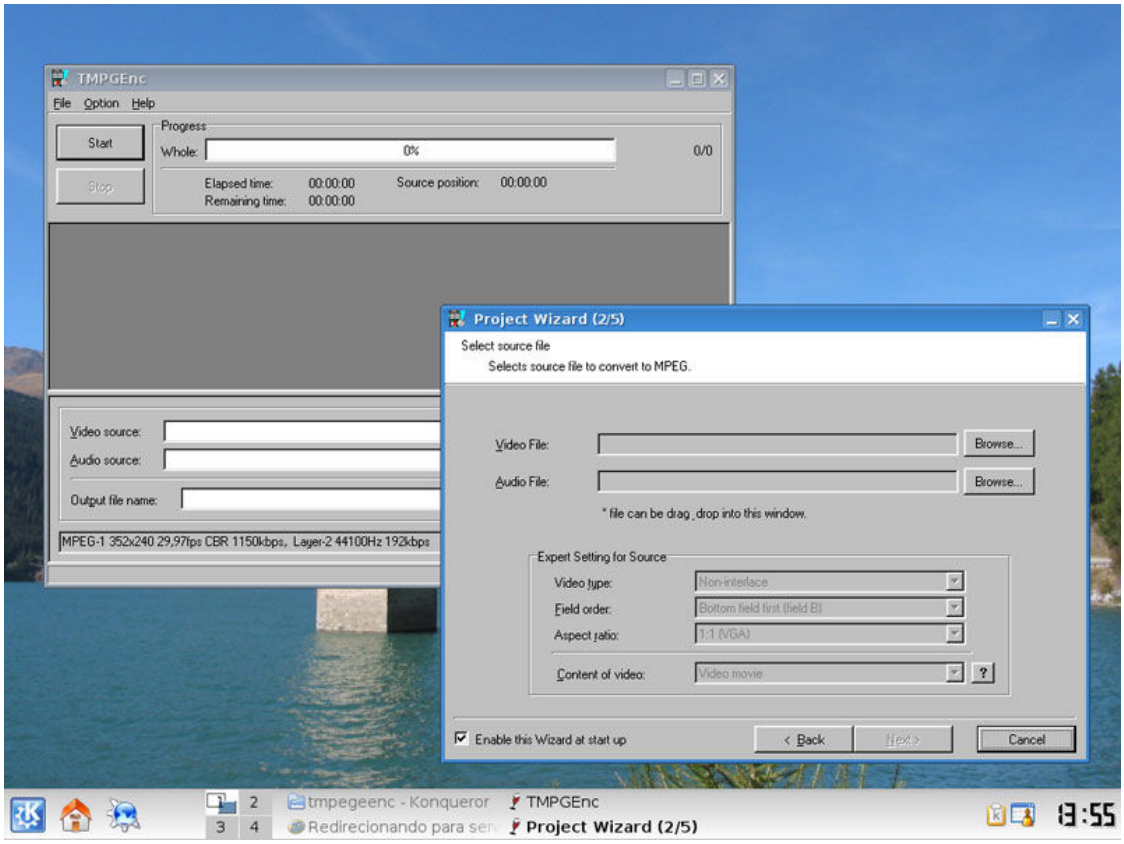


Figura 8.03 - *Software* conversor de vídeo para Windows executando no Linux com o Wine

CONCLUSÃO

A virtualização definitivamente é uma tecnologia emergente. Segundo a revista INFO [INFO249] "80% (oitenta por cento) das grandes e médias empresas brasileiras estão investindo em virtualização e 67% (sessenta e sete por cento) foi o crescimento do mercado mundial de *software* de virtualização em 2005". Ainda segundo a revista INFO [INFO259] "o mercado de virtualização movimentará próximo a 12 (doze) bilhões de dólares até 2011".

A IDC Brasil realizou um estudo intitulado "*Brazil IT Investments Trends 2007*", que foram consultados 100 (cem) organizações do País e revelou que a virtualização está entre as principais prioridades dos CIO (*Chief Information Officer*). Entre os entrevistados, 71% (setenta e um por cento) disseram que vão investir em projetos nessa área em 2008 e pretendem adotar o modelo tanto no ambiente de produção quanto no de teste e desenvolvimento das aplicações.

Neste trabalho pude concluir que a virtualização é praticamente um novo conceito sobre o modelo tradicional da computação, o qual reduz a importância do sistema operacional, permitindo que um *hardware* execute quaisquer aplicações com seu sistema operacional de origem, sem precisar interromper as demais aplicações e serviços já em execução.

Como vimos no capítulo II, a solução de virtualização de sistemas operacionais pode trazer importantes benefícios nos diversos ramos da computação como nas empresas, entidades de pesquisas, instituições de ensino etc.

Há várias formas de virtualização, e cada uma com suas vantagens e desvantagens. Em um processo de implantação é necessário decidir sobre a mais adequada para instituição de modo que se obtenha uma boa relação custo/benefício no processo.

A virtualização nas empresas mesmo sendo muito interessante conforme vimos, como toda nova tecnologia, tem momentos certos para ser adotada e quando adotada, deve ser bem realizada. O mais indicado é que seja traçado um planejamento estratégico adequado, uma vez que sua implantação dispõe de tempo e requer bons conhecimentos

técnicos. Nessa tecnologia não são descartados problemas de compatibilidade e perda de performance, sobretudo por causa de combinações mal realizadas, por exemplo: uma aplicação que consuma muito processamento e memória não deve ser virtualizada junto com outra, com risco de que o desempenho dos dois serviços fique severamente prejudicado. Além disso, as máquinas virtuais tendem a consumir mais recursos (principalmente de CPU e memória), o que torna recomendado dispor de um *hardware* mais robusto a fim de executá-las sem grandes perdas de desempenho.

Também neste trabalho foram citadas características dos principais softwares de virtualização (monitores de máquinas virtuais), além de emuladores e componentes de sistemas operacionais que também se consegue virtualizar. Em alguns também foram feitos testes práticos.

Com essas experiências pude perceber que dos *softwares* monitores de máquinas virtuais, os produtos VMWare e Xen (que são produtos comerciais, mas possuem versões gratuitas) alcançaram um grau de maturidade bastante avançado. No entanto, as demais soluções (Microsoft Virtual Server, VirtualBox, User-mode Linux) por causa de suas particularidades, não deixam nada a desejar e também devem ser consideradas num processo de implantação de virtualização.

Para implantação da virtualização nos ambientes que operam com os sistemas operacionais Solaris e FreeBSD, devem ser considerados suas tecnologias Zones e Jails respectivamente, os quais primam pelo desempenho, uma vez que já estão integrados aos *kernels* desses sistemas. Contudo, nada impede que outras tecnologias sejam aproveitadas.

As tecnologias de emulação total (como QEMU e Bochs) são interessantes quando se deseja um ambiente simples para execução de um sistema operacional sobre o outro e com compatibilidade total. Todavia, por terem que emular quase que totalmente as instruções da CPU, perdem muito em desempenho.

Existem outros sistemas virtuais que neste trabalho não foram citados ou não foram detalhados e que futuramente podem ser explorados em outros trabalhos, são alguns

deles: Denali, Valgrind, Cooperative Linux, Linux VServer, OpenVZ, Virtuozzo, Plex86, Crusoe, Parrot, Cedega e Cygwin.

Uma outra boa sugestão para trabalhos no futuro são testes acerca das novas técnicas de virtualização por meio de uma camada adicional na arquitetura interna dos processadores x86 de última geração (as tecnologias Intel-VT da Intel e AMD-V da AMD).

ANEXO I

TABELA COMPARATIVA DE MÁQUINAS VIRTUAIS

Nome	Criador	Arquitetura CPU Anfitrião	Suporta a Arquitetura de CPU Convidado	SO Anfitriões	Suporta SO como Convidados	Licença	Ezecuta outro SO (*kernel* diferente)	Possui drivers for sistemas convidados suportados	Suporte a SMP para sistemas convidados	Método de operação	Comumente usado por/em	Desempenho do OS convidado em comparação com OS anfitrião	Possui Suporte
Boehs	Kevin Lawton	x86, AMD64, SPARC, PowerPC, Alpha, MIPS	x86, AMD64	Windows, Linux, IRIX, AIX, FreeBSD, OpenBSD, BeOS, Mac OS X	DOS, Windows, xBSD, Linux	LGPL	Sim	?	Sim	Emulação	Desenvolvedores	Muito baixo	?
Containers	Sun Microsystems	x86, x86-64, SPARC	Mesmo que o anfitrião	Solaris 10	Solaris (8 or 10), Linux (BrandZ)	CDDL (Gratuito)	Não	N/D	Sim	Virtualização no nível de sistema operacional	Negócios, desenvolvimento, consolidação de servidores, hospedagem, isolamento de serviços e segurança	Desempenho nativo	Sim
Cooperative Linux	Dan Aloni ajudado por outros desenvolvedores	x86	Mesmo que o pai	Windows NT, 2000, XP, Server 2003, Linux	Linux	GPL versão 2	Não	Alguns	Sim	Virtualização Total	Usado com uma máquina separada para um servidor ou com redes X11	Nativo	?
Denali	Universidade de Washington	x86	x86	Denali	Ilwaco, NetBSD	?	Não	?	Não	Paravirtualização	Pesquisa	Baixo	?
DOSBox	Peter Veenstra and Sjoerd	x86, AMD64, SPARC, PowerPC, Alpha, MIPS	x86	GNU/Linux, Windows, Mac OS Classic, Mac OS X, BeOS, FreeBSD, OpenBSD, Solaris, QNX, IRIX	DOS	GPL	Não	Sim	Não	Emulação com Recompilação Dinâmica	Execução de aplicações DOS, especialmente jogos	Baixo (10% nativo), muito baixo em sistemas não x86	?
DOSEMU	Community Project	x86	x86	Linux	DOS	GPL versão 2	Sim	Sim	Não	Virtualização de Hardware e Software	Suporte a aplicações legadas	Nativo	?
Integrity Virtual Machines	Hewlett-Packard	IA-64	x86-64	HP-UX	HP-UX, Windows, Linux	Comercial	Não	N/D	Sim	Virtualização no nível de sistema operacional	Hospedagem, isolamento de serviços, segurança	Nativo	?
Jail	FreeBSD	x86, AMD64, IA64, ARM, SPARC	Mesmo que o anfitrião	FreeBSD	Mesmo que o anfitrião	Licença FreeBSD	Não	N/D	Sim	Virtualização no nível de sistema operacional	Hospedagem, isolamento de serviços e segurança	Nativo	?
Linux KVM	KVM	CPU Intel/AMD com hardware assist	x86/AMD64	Linux	Linux, Windows	GPL2	Sim	N/D	Sim	Virtualização no nível de sistema operacional	?	Próximo ao nativo	?
Linux-VServer	Projeto Comunitário	x86, AMD64, IA-64, Alpha, PowerPC/64, PA-RISC/64, SPARC/64, ARM, S/390, SH/66, MIPS	Compatível	Linux	Várias distribuições Linux	GPL versão 2	Não	N/D	Sim	Virtualização no nível de sistema operacional	Hospedagem, isolamento de serviços e segurança	Próximo ao nativo	?
Logical Domains	Sun Microsystems	UltraSPARC T1, UltraSPARC T2	Compatível	Solaris	Solaris, Linux, and FreeBSD	?	Não	?	Sim	Paravirtualização	Hospedagem, isolamento de serviços e segurança	Próximo ao nativo	?
OpenVZ	Projeto Comunitário, suportado por Swsoft	Intel x86, AMD64, IA-64, PowerPC64, SPARC/64	Compatível	Linux	Várias distribuições Linux	GPL	Não	Compatível	Sim	Virtualização Total	Isolamento	Próximo ao nativo	?
Parallels Desktop for Mac	Parallels, Inc.	Intel x86; Intel VT-x	x86	Mac OS X (Intel)	Windows, Linux, FreeBSD, OS/2, eComStation, MS-DOS, Solaris	Comercial	Sim	Sim	Não	Virtualização Total	Desenvolvedores, testadores e estações de trabalho	Próximo ao nativo	Sim
Parallels Workstation	Parallels, Inc.	x86; Intel VT-x	x86	Windows, Linux	Windows, Linux, FreeBSD, OS/2, eComStation, MS-DOS, Solaris	Comercial	Sim	Sim	Não	Virtualização Total	Desenvolvedores, testadores e estações de trabalho	Próximo ao nativo	Sim
PearPC	Sebastian Biallas	x86, AMD64, PowerPC	PowerPC	Windows, Linux, Mac OS X, NetBSD	Mac OS X, Darwin, Linux	GPL	Sim	Sim	Não	Emulação com Recompilação Dinâmica	Desenvolvedores, testadores e estações de trabalho	10% mais baixo	?
QEMU	Fabrice Bellard ajudado por outros desenvolvedores	x86, AMD64, IA-64, PowerPC, Alpha, SPARC 32 and 64, ARM, S/390, M68k	x86, AMD64, ARM, SPARC 32 and 64, PowerPC, MIPS	Windows, Linux, Mac OS X, Solaris, FreeBSD, OpenBSD, BeOS	Depende da arquitetura da CPU convidado	GPL/LGPL	Sim	?	Sim	Emulação com Recompilação Dinâmica	Desenvolvedores, testadores e estações de trabalho	10 a 20% mais baixo	?
QuickTransit	Transitive Corp.	AMD64, x86, IA-64, POWER	MIPS, PowerPC, SPARC, x86	Linux, Mac OS X, Solaris	Linux, Mac OS X, Irix, Solaris	Comercial	Sim	Não	Sim	Emulação com Recompilação Dinâmica	Vários	Varia dependendo da combinação anfitrião/convidado	Sim

Nome	Criador	Arquitetura CPU Anfitrião	Suporta a Arquitetura de CPU Convidado	SO Anfitriões	Suporta SO como Convidados	Licença	Executa outro SO (*kernel* diferente)	Possui drivers for sistemas convidados suportados	Suporte a SMP para sistemas convidados	Método de operação	Comumente usado por/em	Desempenho do OS convidado em comparação com OS anfitrião	Possui Suporte
SimNow	AMD	AMD64	AMD64	Linux (64 bit), Windows (64 bit)	Linux, Windows (32bit and 64bit)	Comercial	Sim	Sim	Sim	Virtualização Total	Desenvolvedores e servidores	10 vezes mais baixo	?
User Mode Linux	Jeff Dike ajudado por outros desenvolvedores	x86, x86-64, PowerPC	Mesmo que o pai	Linux	Linux	GPL versão 2	Não	Módulos especiais requeridos	Sim	Virtualização Total	Usado com uma máquina separada para um servidor ou com redes X11	Próximo ao nativo	?
VDSmanager	ISPsystem LLC	x86	Mesmo que o anfitrião	FreeBSD	FreeBSD	Comercial	Não	N/D	Sim	Virtualização no nível de sistema operacional	Hospedagem, isolamento de serviços e segurança	Próximo ao nativo	Sim
VirtualBox	InnoTek	x86, x86-64	x86	Windows, Linux, Mac OS X	DOS, Windows, Linux, OS/2, FreeBSD	GPL V2; versão completa comercial	Sim	Sim	Não	Virtualização Total	Estações de trabalho, consolidação de servidores e desenvolvimento	Próximo ao nativo	Sim (com licença comercial)
Virtual PC 2007	Microsoft	x86, x64	x86	Windows Vista (Business, Enterprise, Ultimate), XP Pro, XP Tablet PC Edition	DOS, Windows, OS/2	Gratuito	Sim	Sim	Não	Virtualização Total	Estações de trabalho, consolidação de servidores e desenvolvimento	Próximo ao nativo	?
Virtual Server	Microsoft	Intel x86, AMD64	Intel x86	Windows 2003, XP	Windows NT, 2000, 2003, Linux (Red Hat e SUSE)	Gratuita	Sim	Sim	Não	Virtualização Total	Parque de servidores	Próximo ao nativo	?
Virtuozzo	SWsoft	x86, IA-64, AMD64	x86, IA-64, AMD64	Linux & Windows	Várias distribuições Linux, Windows	Comercial	Não	Compatível	Sim	Virtualização no nível de sistema operacional	Consolidação de servidores, recuperação de desastres e provedores de serviço	Próximo ao nativo	Sim
VMware ESX Server	VMware	x86, AMD64	x86, AMD64	SO próprio	Windows, Red Hat, SuSE, Netware, Solaris, FreeBSD	Comercial			Sim	Virtualização Total	Consolidação de servidores	Próximo ao nativo	
VMware Server	VMware	x86, AMD64	x86, AMD64	Windows, Linux	DOS, Windows, Linux, FreeBSD, Netware, Solaris, Virtual appliances	Gratuita	Sim	Sim	Sim	Virtualização Total	Servidores, desenvolvedores e testadores	Próximo ao nativo com perda de desempenho com o aumento de carga	Sim
VMware Workstation	VMware	x86, AMD64	x86, AMD64	Windows, Linux	DOS, Windows, Linux, FreeBSD, Netware, Solaris, Virtual appliances	Comercial	Sim	Sim	Sim	Virtualização Total	Profisionais técnicos e desenvolvedores avançados	Próximo ao nativo	Sim
VMware Player	VMware	x86, AMD64	x86, AMD64	Windows, Linux	DOS, Windows, Linux, FreeBSD, Netware, Solaris, Virtual appliances	Gratuita	Sim	Sim	Não	Virtualização Total	Profisionais técnicos, desenvolvedores avançados e usuários finais	Próximo ao nativo com perda de desempenho com o aumento de carga	Sim
Xen	Universidade de Cambridge, Intel e AMD	x86, AMD64, (PowerPC e x86-64 em progresso)	Mesmo que o anfitrião	NetBSD, Linux, Solaris	Linux, NetBSD, FreeBSD, OpenBSD, Solaris, Windows XP & 2003 Server (necessário CPU com Intel-VT ou AMD-V)	GPL	Sim	Não requerido com exceção dos *drivers* de rede onde um NAT é requerido. Requer um *kernel* convidado modificado ou nível de abstração de *hardware* .	Sim	Paravirtualização	?	Próximo ao nativo com perda de desempenho com o aumento de carga	Sim
z/VM	IBM	Arquitetura z/	Arquitetura z/	Nenhum	Linux on zSeries, z/OS, z/VSE, z/TPF, z/VM, VM/CMS, MUSIC/SP, e predecessores	Comercial	Sim	Sim, mas não requerido	Sim	Virtualização de Hardware e Software	Servidores de empresas	Variável	?

BIBLIOGRAFIA

[ADAMS e AGESEN] ADAMS, K. and AGESEN, O. A Comparison of *Software* and *Hardware* Techniques for x86 Virtualization, VMWare.

[BARHAM et al] BARHAM, P., DRAGOVIC B., FRASER K., HAND, S., HARRIS T., HO, A., NEUGEBAUER, R., PRATT, I. and WARFIELD, A. Xen and the Art of Virtualization, In Proceedings of the 19th ACM Symposium on Operating Systems Principles, pages 164–177, Bolton Landing, NY, October 2003.

[BELLARD 2005] BELLARD, F. QEMU, A Fast and Portable Dynamic Translator, USENIX Annual Technical Conference, USENIX Association, 2005.

[BELLARD 2007] BELLARD, F. QEMU Emulator User Documentation. February 2007. Disponível em <http://fabrice.bellard.free.fr/qemu/qemu-doc.html>.

[BOCCATO et al] BOCCATO, B., DIAS R., CRUZ, R., Máquina Virtual .NET. Disponível em <www.ic.unicamp.br/~rodolfo/Cursos/mc722/2s2005/Trabalho/g06-net.pdf>.

[CLARK] CLARK, C. Xen v3.0 Users' Manual, 2002. <Disponível em: http://www.cl.cam.ac.uk/research/srg/netos/xen/readmes/user.pdf>.

[DIKE] DIKE, Jeff. A User-mode Linux. <Disponível em: http://lwn.net/2001/ features/OLS/pdf/pdf/uml.pdf>

[DUMIENSE e JESUS] DUMIENSE, G. M, JESUS, J. P. Virtualização e Segurança em Processadores Modernos, IST, 2006/2007.

[HIGASHIYAMA] HIGASHIYAMA, A. Visão Comercial do Microsoft Virtual Server 2005 R2, Artigo Técnico, Revista IT Central, Junho 2006.

[HINES] HINES, Matt., Research Points to Faster Threat Development, eWeek, 2006.

[INFO249] INFO EXAME, Revista, Editora Abril S.A., Edição nº 249, Dezembro 2006.

[INFO259] INFO EXAME, Revista, Editora Abril S.A, Edição nº 259, Outubro 2007.

[INTEL] INTEL, Using virtualization to change the face of business - E-book 2006. Disponível em: <http://intelvt.com/flash>.

[KAMP e WATSON] KAMP, P., WATSON, R., Jails: Confining the omnipotent root. The FreeBSD Project. 2nd International System Administration and Networking Conference "SANE 2000", May 2000.

[KASICK et al] KASICK, M., WILLEN, G. and CUI, M. Virtualization - Operating System Design & Implementation, April 2007.

[LAUREANO 2004] LAUREANO, M. Uma abordagem para a proteção de detectores de intrusão baseada em máquinas virtuais. Centro de Ciências Exatas e de Tecnologia, Pontifícia Universidade Católica do Paraná, Curitiba, 2004. 103 f.

[LAUREANO 2006] LAUREANO, M. Máquinas Virtuais e Emuladores. Conceitos, Técnicas e Aplicações. Ed. Novatec 2006.

[LAWTON et al] LAWTON, K. P., DENNEY, B., GUARNERI, D. N., RUPPERT, V., BOTHAMY, C. Bochs User Manual. Disponível em <http://bochs.sourceforge.net>

[LINDHOLM e YELLIN] LINDHOLM, T., YELLIN F. The Java Virtual Machine Specification, Second Edition, Sun Microsystems, Inc. 1999.

[LINUX] LINUX MAGAZINE, Revista. Linux New Media do Brasil Editora Ltda., Edição nº 9, Junho de 2005.

[MAGENHEIMER e CHRISTIAN] MAGENHEIMER D. and CHRISTIAN T., vBlades: Optimized Paravirtualization for the Itanium Processor Family. Proceedings of the Third Virtual Machine Research and Technology Symposium, Usenix, May 2004.

[MICROSOFT] MICROSOFT. Windows Server Virtualization – An Overview, Microsoft Corporation, May 2006.

[POPEK e GOLDBERG] POPEK, G. J., and GOLDBERG, R. P. Formal requirements for virtualizable third generation architectures. ACM 17, 7 (1974).

[PRZYBYSZ e LUIZ] PRZYBYSZ, A. L. e LUIZ Jr., O. J. Sistema Operacional Xen e o modelo VM Monitor, Mestrado em Informática Aplicada, PUCPR, Março 2006.

[ROSENBLUM] ROSENBLUM, M. The Reincarnation of Virtual Machines. ACM Queue vol. 2, no. 5 - July/August 2004.

[SINGH] SINGH, An Introduction to Virtualization. Disponível em: <http://www.*kernel*thread.com/publications/virtualization>.

[SUN 2003] SUN Microsystems Inc., Sun Ray Interoperability Brief, White Paper, August 2003.

[SUN 2004] SUN Microsystems Inc., System Administration Guide: Solaris Containers, Resource Management, and Zones, March 2004.

[SYMANTEC] SYMANTEC Corporation, Symantec Internet Security Threat Report, 2006.

[STEIL] STEIL M. Inside VMWare - How VMware, VirtualPC and Parallels actually work. 23rd Chaos Communication Congress.

[VICTOR] VICTOR, J. Solaris Containers Techonology Architecture Guide. Sun Microsystems. Sun BluePrints OnLine, May 2006.

[VIRTUALBOX] VIRTUALBOX. Technical Documentations, Innotek GmbH, 2007. Disponível em <http://www.virtualbox.org>

[VMWARE 2006-A] VMWARE Inc., Architeture and Performance Implications. VMware ESX Server 2. 2006. White paper.

[VMWARE 2006-B] VMWARE Inc., VMware Infrastrucutre Architecture Overview, 2006 White paper.

[WHATELY e AMORIM] WHATELY L. e AMORIM, C. Sistemas de Computação Baseados em Máquinas Virtuais. Engenharia de Sistemas e Computação - COPPE/UFRJ. 2005.

[WINEHQ] WINEHQ, Wine Developer's Guide, Wine Project Official Page. Disponível em <http://www.winehq.org/*site*/docs/winedev-guide/index>.

[YEHUDA] YEHUDA M.,User Mode Linux. Study Group, HRL. IBM Haifa Research Labs., October 2003.

[YOUSEFF et al] YOUSEFF, L., WOLSKI, R. GORDA, B. and KRINTZ, C. Paravrtualization for HPC Systems. Department of Computer Science. University of California, Santa Barbara.